Docket No. 240554US2/shb

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Kunihiro AKIYOSHI, et al. GAU: 2622

SERIAL NO: 10/627,731 EXAMINER:

FILED: July 28, 2003

FOR: IMAGE FORMING APPARATUS, INFORMATION PROCESSING APPARATUS AND VERSION CHECK METHOD

REQUEST FOR PRIORITY

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §120.**

☐ Full benefit of the filing date(s) of U.S. Provisional Application(s) is claimed pursuant to the provisions of **35 U.S.C. §119(e):** **Application No.** **Date Filed**

■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
Japan	2002-224136	July 31, 2002
Japan	2003-195193	July 10, 2003
Japan	2003-195194	July 10, 2003

Certified copies of the corresponding Convention Application(s)

■ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

☐ (B) Application Serial No.(s)

☐ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Joseph A. Scafetta Jr.

Marvin J. Spivak
Registration No. 24,913
Joseph A. Scafetta, Jr.
Registration No. 26,803

Customer Number
22850

Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 05/03)

10/627,731

65

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年　７月３１日

出願番号
Application Number: 特願２００２－２２４１３６

[ST.10/C]: [ＪＰ２００２－２２４１３６]

出願人
Applicant(s): 株式会社リコー

２００３年　５月　６日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３０３２４４１

【書類名】　特許願
【整理番号】　0203940
【提出日】　平成14年 7月31日
【あて先】　特許庁長官殿
【国際特許分類】　G03G 21/00　370
【発明の名称】　画像形成装置およびバージョンチェック方法
【請求項の数】　27
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　秋吉　邦洋
【発明者】
　【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
　【氏名】　安藤　光男
【特許出願人】
　【識別番号】　000006747
　【氏名又は名称】　株式会社リコー
【代理人】
　【識別番号】　100089118
　【弁理士】
　【氏名又は名称】　酒井　宏明
【手数料の表示】
　【予納台帳番号】　036711
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9808514
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　画像形成装置およびバージョンチェック方法

【特許請求の範囲】

【請求項１】　画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションと、前記アプリケーションと画像形成処理で使用されるハードウェア資源との間に介在し、前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスとを備えた画像形成装置であって、

前記アプリケーションのインストール時から起動後の実行時までの間に、前記アプリケーションが使用する関数単位毎のバージョン情報を取得すると共に、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得し、対応する関数のバージョンが同じか否かをチェックする関数単位バージョンチェック手段

を備えたことを特徴とする画像形成装置。

【請求項２】　前記仮想アプリケーションサービスは、前記アプリケーション毎に設けられ、前記各アプリケーションを個別に仮起動または通常起動させることを特徴とする請求項１に記載の画像形成装置。

【請求項３】　前記コントロールサービスは、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うことを特徴とする請求項１または２に記載の画像形成装置。

【請求項４】　前記関数単位バージョンチェック手段による関数単位毎のバージョンチェックを行う前に、前記各アプリケーションの全体バージョン情報を取得すると共に、前記仮想アプリケーションサービスの全体バージョン情報を取得し、全体バージョンが同じか否かをチェックする全体バージョンチェック手段をさらに備え、

前記全体バージョンチェック手段による比較結果に基づいて全体バージョンが

異なっている場合のみ、前記関数単位バージョンチェック手段による関数単位毎のバージョンチェックを行うことを特徴とする請求項１～３のいずれか一つに記載の画像形成装置。

【請求項５】　前記アプリケーションが使用する関数単位毎のバージョン情報と前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報、および、前記アプリケーションの全体バージョン情報と前記仮想アプリケーションサービスの全体バージョン情報の少なくとも一方のバージョン情報を取得するバージョン情報取得手段をさらに備えたことを特徴とする請求項１～４のいずれか一つに記載の画像形成装置。

【請求項６】　前記バージョン情報取得手段は、予め必要なバージョン情報を持たせた実行ファイルを用いて、前記アプリケーションの仮起動、あるいは、通常起動の際にプロセス間通信を行うことにより、所望のバージョン情報を取得することを特徴とする請求項５に記載の画像形成装置。

【請求項７】　前記バージョン情報取得手段は、前記アプリケーションの実行ファイルを生成する際に、前記アプリケーションが使用する関数単位毎のバージョン情報を記載した使用関数テーブルを生成すると共に、前記仮想アプリケーションサービスの実行ファイルを生成する際に、前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を記載した全関数テーブルを生成し、前記使用関数テーブルと前記全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報を取得することを特徴とする請求項５に記載の画像形成装置。

【請求項８】　前記関数単位バージョンチェック手段は、前記アプリケーションのインストール時、前記アプリケーションの起動時、および、前記アプリケーションの起動後の実行時のいずれかの時点でバージョンチェックを行うことを特徴とする請求項１～７のいずれか一つに記載の画像形成装置。

【請求項９】　前記関数単位バージョンチェック手段が前記アプリケーションのインストール時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを仮起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする

請求項８に記載の画像形成装置。

【請求項１０】　前記関数単位バージョンチェック手段が前記アプリケーションの起動時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを通常起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする請求項８に記載の画像形成装置。

【請求項１１】　前記関数単位バージョンチェック手段が前記アプリケーションの起動後の実行時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを通常起動させた際に取得した、当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックすることを特徴とする請求項８に記載の画像形成装置。

【請求項１２】　前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記仮想アプリケーションサービスに含まれていることを特徴とする請求項１～１１のいずれか一つに記載の画像形成装置。

【請求項１３】　前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記仮想アプリケーションサービスのプロセス内部でスレッドとして生成されることを特徴とする請求項１２に記載の画像形成装置。

【請求項１４】　前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記アプリケーションに含まれることを特徴とする請求項１～１１のいずれか一つに記載の画像形成装置。

【請求項１５】　前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記アプリケーションのプロセス内部でスレッドとして生成されることを特徴とする請求項１４に記載の画像形成装置。

【請求項１６】　画像形成処理にかかるユーザサービスにそれぞれ固有の処

理を行うアプリケーションと、前記アプリケーションと画像形成処理で使用されるハードウェア資源との間に介在し、前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスとを備えた画像形成装置上で、前記アプリケーションとの整合性をチェックするバージョンチェック方法であって、

前記アプリケーションのインストール時から起動後の実行時までの間に、前記アプリケーションが使用する関数単位毎のバージョン情報を取得すると共に、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得し、対応する関数のバージョンが同じか否かをチェックする関数単位バージョンチェックステップ

を含むことを特徴とするバージョンチェック方法。

【請求項１７】　前記コントロールサービスは、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うことを特徴とする請求項１６に記載のバージョンチェック方法。

【請求項１８】　前記関数単位バージョンチェックステップによる関数単位毎のバージョンチェックを行う前に、前記各アプリケーションの全体バージョン情報を取得すると共に、前記仮想アプリケーションサービスの全体バージョン情報を取得し、全体バージョンが同じか否かをチェックする全体バージョンチェックステップをさらに含み、

前記全体バージョンチェックステップによる比較結果に基づいて全体バージョンが異なっている場合のみ、前記関数単位バージョンチェックステップによる関数単位毎のバージョンチェックを行うことを特徴とする請求項１６または１７に記載のバージョンチェック方法。

【請求項１９】　前記アプリケーションが使用する関数単位毎のバージョン情報と前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報、および、前記アプリケーションの全体バージョン情報と前記仮想

アプリケーションサービスの全体バージョン情報の少なくとも一方のバージョン情報を取得するバージョン情報取得ステップをさらに含むことを特徴とする請求項１６～１８のいずれか一つに記載のバージョンチェック方法。

【請求項２０】　前記バージョン情報取得ステップは、予め必要なバージョン情報を持たせた実行ファイルを用いて、前記アプリケーションの仮起動、あるいは、通常起動の際にプロセス間通信を行うことにより、所望のバージョン情報を取得することを特徴とする請求項１９に記載のバージョンチェック方法。

【請求項２１】　前記バージョン情報取得ステップは、前記アプリケーションの実行ファイルを生成する際に、前記アプリケーションが使用する関数単位毎のバージョン情報を記載した使用関数テーブルを生成すると共に、前記仮想アプリケーションサービスの実行ファイルを生成する際に、前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を記載した全関数テーブルを生成し、前記使用関数テーブルと前記全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報を取得することを特徴とする請求項１９に記載のバージョンチェック方法。

【請求項２２】　前記関数単位バージョンチェックステップは、前記アプリケーションのインストール時、前記アプリケーションの起動時、および、前記アプリケーションの起動後の実行時のいずれかの時点でバージョンチェックを行うことを特徴とする請求項１６～２１のいずれか一つに記載のバージョンチェック方法。

【請求項２３】　前記関数単位バージョンチェックステップで前記アプリケーションのインストール時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを仮起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする請求項２２に記載のバージョンチェック方法。

【請求項２４】　前記関数単位バージョンチェックステップで前記アプリケーションの起動時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを通常起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする

請求項２２に記載のバージョンチェック方法。

【請求項２５】　前記関数単位バージョンチェックステップで前記アプリケーションの起動後の実行時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを通常起動させた際に取得した、当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックすることを特徴とする請求項２２に記載のバージョンチェック方法。

【請求項２６】　前記関数単位バージョンチェックステップ、前記全体バージョンチェックステップ、および、前記バージョン情報取得ステップの少なくとも１つの処理が前記仮想アプリケーションサービスで行われることを特徴とする請求項１６～２５のいずれか一つに記載のバージョンチェック方法。

【請求項２７】　前記関数単位バージョンチェックステップ、前記全体バージョンチェックステップ、および、前記バージョン情報取得ステップの少なくとも１つの処理が前記アプリケーションで行われることを特徴とする請求項１６～２５のいずれか一つに記載のバージョンチェック方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、コピー、プリンタ、スキャナおよびファクシミリなどの画像形成処理にかかるユーザサービスを提供するアプリケーションのバージョン不整合による不具合発生を未然に防止することができる画像形成装置およびバージョンチェック方法に関するものである。

【０００２】

【従来の技術】

近年では、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を１つの筐体内に収納した画像形成装置（以下、「複合機」という。）が知られている。この複合機は、１つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応した３種類のソフトウェアを設け、これらのソフトウェアを切り替えることによって、当該

装置をプリンタ、コピー、スキャナまたはファクシミリ装置として動作させるものである。

【０００３】

このような従来の複合機では、プリンタ、コピー、ファクシミリ、スキャナなどの各機能単位でアプリケーションプログラムが起動され、ハードウェア資源にアクセスする機能を持ち合わせている。その際、アプリケーションプログラムとオペレーティングシステム（ＯＳ）のバージョンが同じことが前提となるが、例えば、ＯＳがバージョンアップしてアプリケーションとの間でバージョン差が生じた場合、今まで使えていた機能が使えなくなったり、アプリケーションそのものが起動しなくなったりすることがある。

【０００４】

このため、従来の複合機では、ＯＳがバージョンアップされた場合、それに伴ってアプリケーションをコンパイルし直すなどして、常に整合性のとれたバージョン関係にあることが要請されている。

【０００５】

【発明が解決しようとする課題】

ところで、このような従来の複合機では、プリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェアをそれぞれ別個に設けているため、各ソフトウェアの開発に多大の時間を要する。このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピーまたはファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットホームを備えた画像形成装置（複合機）を発明した。

【０００６】

このような新規な複合機では、アプリケーションとコントロールサービスが別個に設けられているため、複合機の出荷後にユーザもしくは第三者であるサード

ベンダが新規なアプリケーションを開発して複合機に搭載することが可能であり、これによって多種多様な機能を提供することが可能となっている。

【０００７】

このような新規な複合機では、アプリケーションの少なくとも２つが共通的に必要とするサービスを提供するコントロールサービスがアプリケーションと別個に設けられているので、新規アプリケーションを開発する場合、各種コントロールサービスとのプロセス間通信を実現する処理をソースコードで記述する必要がある。しかしながら、新規アプリケーションを開発する場合は、各コントロールサービスが提供する関数やメッセージなどを正確に把握した上で、予め規定された手順で記述しなければならず、各コントロールサービスやアプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）がデバッグや機能追加によってバージョンアップが繰り返されると、ベンダーはどのバージョンに合わせてアプリケーション開発をすればよいかが非常に分かり難くなり、アプリケーションの開発自体が阻害されるおそれが出てきた。このことは、固定的な複数の機能を寄せ集めた従来の複合機では問題にならなかった新規な課題である。

【０００８】

また、各コントロールサービスとアプリケーションのインタフェース（ＡＰＩ）のすべてを、新規アプリケーションを開発するサードベンダなどの第三者に開示することは、プログラムの秘匿性の面から言って好ましくない。特に、複合機のシステムに大きく影響を与えるような箇所については、第三者に対して隠蔽することにより、直接コントロールサービスに対してアクセスされるのを回避することが可能となり、複合機のセキュリティ面および障害発生の未然防止の面からも好ましい。

【０００９】

そこで、出願人は、上記複合機のコントロールサービスとアプリケーションとの間に一定のサポート範囲でバージョン差を吸収して整合性をとると共に、アプリケーションとコントロールサービスとの間の複雑なインタフェースを隠蔽して重要なインタフェースの秘匿性を確保しつつ、特定のインタフェースを開示して

新規アプリケーションの開発効率を向上させることのできる仮想アプリケーションサービス（ＶＡＳ）を追加したさらに新規な複合機を発明した。この仮想アプリケーションサービス（ＶＡＳ）は、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ、アプリケーションをクライアントとしたサーバプロセスとして動作するものである。

【００１０】

しかしながら、このような新規な複合機における仮想アプリケーションサービス（ＶＡＳ）のバージョンは、常にコントロールサービス側と整合性がとれるようにバージョンアップが繰り返されると、今度はアプリケーションと仮想アプリケーションサービス（ＶＡＳ）との間でバージョンの不整合が生じ易くなり、実行時（ランタイム時）におけるエラーの発生が問題となる可能性が出てきた。

【００１１】

そこで、アプリケーションと仮想アプリケーションサービス（ＶＡＳ）のバージョンの整合性がとれているか否かを事前にチェックして、ランタイム時におけるエラー発生を防止するため、全体のバージョン同士を比較して、一致していれば整合性有りとし、一致していない場合は整合性無しとすることも可能である。しかし、これでは僅かなバージョン差が生じても整合性無しとされるため、バージョン差の吸収範囲があまりにも狭く、コントロールサービス側でバージョンアップがあると使用できなくなるアプリケーションが多数発生するおそれがあった。

【００１２】

この発明は上記に鑑みてなされたもので、コントロールサービス側のバージョンアップに伴って仮想アプリケーションサービスがバージョンアップされても、アプリケーションとの間に生じたバージョン差をできるだけ吸収して、利用可能なアプリケーションを増やすと共に、アプリケーションのランタイム時におけるエラーの発生を確実に防止することができる画像形成装置およびバージョンチェック方法を得ることを目的とする。

【００１３】

【課題を解決するための手段】

上記目的を達成するため、請求項１にかかる発明は、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションと、前記アプリケーションと画像形成処理で使用されるハードウェア資源との間に介在し、前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスとを備えた画像形成装置であって、前記アプリケーションのインストール時から起動後の実行時までの間に、前記アプリケーションが使用する関数単位毎のバージョン情報を取得すると共に、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得し、対応する関数のバージョンが同じか否かをチェックする関数単位バージョンチェック手段を備えたことを特徴とする。

【００１４】

この請求項１にかかる発明によれば、関数単位バージョンチェック手段によって、アプリケーションのインストール時から起動後の実行時までの間に、アプリケーションが使用する関数単位毎のバージョン情報を取得すると共に、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得して、対応する関数のバージョンが同じか否かをチェックするようにしたため、アプリケーションが使用していない関数に対応した仮想アプリケーションサービスのインターフェイスの関数が変更、追加、あるいは削除されてバージョンアップしたとしても、両者間の整合性に影響を与えないことから、全体バージョン同士を単純比較する場合と比べて、バージョン差の吸収範囲が広くなり、利用可能なアプリケーションを増やすことができる。また、アプリケーションが実際に使用する関数単位毎のバージョン同士を個別に比較するため、確実にランタイム時におけるエラー発生を防止することができる。

【００１５】

また、請求項２にかかる発明は、請求項１に記載の画像形成装置において、前

記仮想アプリケーションサービスは、前記アプリケーション毎に設けられ、前記各アプリケーションを個別に仮起動または通常起動させることを特徴とする。

【００１６】

この請求項２にかかる発明によれば、仮想アプリケーションサービスは、アプリケーション毎に設けられ、各アプリケーションを個別に仮起動または通常起動させるようにしたため、各アプリケーションが使用する関数単位毎のバージョン情報を取得し、これに対応した仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得して、対応する関数同士のバージョンチェックを容易に行うことができる。

【００１７】

また、請求項３にかかる発明は、請求項１または２に記載の画像形成装置において、前記コントロールサービスは、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うことを特徴とする。

【００１８】

この請求項３にかかる発明によれば、コントロールサービスは、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うようにしたため、複数のアプリケーションを作成する際に、共通化できる部分をコントロールサービス（プラットホーム）として共通化することにより、それ以外の部分を作成すればよく、アプリケーションの開発が容易となり、かつ、短期間で開発することができる。

【００１９】

また、請求項４にかかる発明は、請求項１～３のいずれか一つに記載の画像形成装置において、前記関数単位バージョンチェック手段による関数単位毎のバージョンチェックを行う前に、前記各アプリケーションの全体バージョン情報を取得すると共に、前記仮想アプリケーションサービスの全体バージョン情報を取得し、全体バージョンが同じか否かをチェックする全体バージョンチェック手段をさらに備え、前記全体バージョンチェック手段による比較結果に基づいて全体バ

ージョンが異なっている場合のみ、前記関数単位バージョンチェック手段による関数単位毎のバージョンチェックを行うことを特徴とする。

【００２０】

この請求項４にかかる発明によれば、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報とを取得して、アプリケーションと仮想アプリケーションサービスの全体バージョン同士を比較する全体バージョンチェック手段をさらに備えていて、全体バージョンが異なっている場合にのみ関数単位バージョンチェック手段による関数単位毎のバージョンチェックを行うようにしたため、アプリケーションが使用する関数単位毎の個別のバージョンチェックを常に行う必要がなくなり、バージョンチェック処理を効率良く、かつ迅速に行うことができる。

【００２１】

また、請求項５にかかる発明は、請求項１～４のいずれか一つに記載の画像形成装置において、前記アプリケーションが使用する関数単位毎のバージョン情報と前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報、および、前記アプリケーションの全体バージョン情報と前記仮想アプリケーションサービスの全体バージョン情報の少なくとも一方のバージョン情報を取得するバージョン情報取得手段をさらに備えたことを特徴とする。

【００２２】

この請求項５にかかる発明によれば、バージョン情報取得手段により、アプリケーションが使用する関数単位毎のバージョン情報と仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得したり、あるいは、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報を取得することができるため、取得したバージョン情報同士を比較するだけで、容易にバージョンチェックを行うことができる。

【００２３】

また、請求項６にかかる発明は、請求項５に記載の画像形成装置において、前記バージョン情報取得手段は、予め必要なバージョン情報を持たせた実行ファイルを用いて、前記アプリケーションの仮起動、あるいは、通常起動の際にプロセ

ス間通信を行うことにより、所望のバージョン情報を取得することを特徴とする。

【００２４】

この請求項６にかかる発明によれば、バージョン情報取得手段は、予め必要なバージョン情報を持たせた実行ファイルを使って、アプリケーションを仮起動、あるいは、通常起動させた際のプロセス間通信により、所望のバージョン情報を取得するようにしたため、バージョン情報を容易に一元管理することができる。

【００２５】

また、請求項７にかかる発明は、請求項５に記載の画像形成装置において、前記バージョン情報取得手段は、前記アプリケーションの実行ファイルを生成する際に、前記アプリケーションが使用する関数単位毎のバージョン情報を記載した使用関数テーブルを生成すると共に、前記仮想アプリケーションサービスの実行ファイルを生成する際に、前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を記載した全関数テーブルを生成し、前記使用関数テーブルと前記全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報を取得することを特徴とする。

【００２６】

この請求項７にかかる発明によれば、バージョン情報取得手段は、アプリケーションの実行ファイルの生成時にアプリケーションが使用する関数毎のバージョン情報が記載された使用関数テーブルを生成し、仮想アプリケーションサービスの実行ファイルの生成時に仮想アプリケーションサービスのインターフェイスの関数毎のバージョン情報が記載された全関数テーブルを生成して、使用関数テーブルと全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報が取得できるため、アプリケーションを仮起動や通常起動させることなくバージョンチェックを行うことができる。

【００２７】

また、請求項８にかかる発明は、請求項１～７のいずれか一つに記載の画像形成装置において、前記関数単位バージョンチェック手段は、前記アプリケーションのインストール時、前記アプリケーションの起動時、および、前記アプリケー

ションの起動後の実行時のいずれかの時点でバージョンチェックを行うことを特徴とする。

【００２８】

この請求項８にかかる発明によれば、関数単位バージョンチェック手段は、アプリケーションのインストール時、アプリケーションの起動時、および、アプリケーションの起動後の実行時のいずれかの時点でバージョンチェックを行うようにしたため、アプリケーションのインストール状況等に応じて関数単位毎のバージョンチェックを適切な時期に行うことが可能となり、アプリケーションのバージョン不整合によるランタイム時のエラー発生を未然に防ぐことができる。

【００２９】

また、請求項９にかかる発明は、請求項８に記載の画像形成装置において、前記関数単位バージョンチェック手段が前記アプリケーションのインストール時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを仮起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする。

【００３０】

この請求項９にかかる発明によれば、関数単位バージョンチェック手段がアプリケーションのインストール時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを仮起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるため、未インストールの新規アプリケーションをインストールする際に、仮想アプリケーションサービスとの間のバージョンの整合性をチェックすることができる。このように、インストールされていないアプリケーションであっても仮起動させることにより、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信が可能となるため、各種情報を双方向でやり取りすることが可能となる。

【００３１】

また、請求項１０にかかる発明は、請求項８に記載の画像形成装置において、前記関数単位バージョンチェック手段が前記アプリケーションの起動時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケ

ーションを通常起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする。

【００３２】

この請求項１０にかかる発明によれば、関数単位バージョンチェック手段がアプリケーションの起動時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを通常起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるため、インストール済みのアプリケーション、あるいは、未インストールのアプリケーションをとりあえずインストールしておいて、通常起動の際にバージョンチェックを受けることができる。この場合は、アプリケーションを通常起動させるため、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信を行うことが可能であり、各種情報を双方向でやり取りすることができる。

【００３３】

また、請求項１１にかかる発明は、請求項８に記載の画像形成装置において、前記関数単位バージョンチェック手段が前記アプリケーションの起動後の実行時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを通常起動させた際に取得した、当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックすることを特徴とする。

【００３４】

この請求項１１にかかる発明によれば、関数単位バージョンチェック手段がアプリケーションの起動後の実行時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを通常起動させた際に取得した当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックするようにしたため、アプリケーションの起動後の実行時においてバージョンチェックを受けることができる。この場合は、既にアプリケーションを通常起動させているため、アプリケーションと仮想アプリケーションサービスとの間でのプロセス間通信が可能であり、各種情報を双方向でやり取りすることができる。

【００３５】

また、請求項１２にかかる発明は、請求項１～１１のいずれか一つに記載の画像形成装置において、前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記仮想アプリケーションサービスに含まれていることを特徴とする。

【００３６】

この請求項１２にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つが仮想アプリケーションサービスに含まれているため、バージョンチェック処理を主に仮想アプリケーションサービス側で行うことができる。

【００３７】

また、請求項１３にかかる発明は、請求項１２に記載の画像形成装置において、前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記仮想アプリケーションサービスのプロセス内部でスレッドとして生成されることを特徴とする。

【００３８】

この請求項１３にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つが仮想アプリケーションサービスのプロセス内部でスレッドとして生成されるため、複数のアプリケーションが並列に起動されたり、バージョン情報取得処理と関数単位バージョンチェック処理と全体バージョンチェック処理とを、コンテクスト切り替えなしにＣＰＵ占有切り替えによる並列実行を行うことが可能となり、これらの並列処理を円滑に行うことができる。

【００３９】

また、請求項１４にかかる発明は、請求項１～１１のいずれか一つに記載の画像形成装置において、前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記アプリケーションに含まれることを特徴とする。

【００４０】

この請求項１４にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つがアプリケーションに含まれているため、バージョンチェック処理を主にアプリケーション側で行うことができる。

【００４１】

また、請求項１５にかかる発明は、請求項１４に記載の画像形成装置において、前記関数単位バージョンチェック手段、前記全体バージョンチェック手段、および、前記バージョン情報取得手段の少なくとも１つは、前記アプリケーションのプロセス内部でスレッドとして生成されることを特徴とする。

【００４２】

この請求項１５にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つがアプリケーションのプロセス内部でスレッドとして生成されるため、バージョン情報取得処理と関数単位バージョンチェック処理と全体バージョンチェック処理とを、コンテクスト切り替えなしにＣＰＵ占有切り替えによる並列実行を行うことが可能となり、これらの並列処理を円滑に行うことができる。

【００４３】

また、請求項１６にかかる発明は、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションと、前記アプリケーションと画像形成処理で使用されるハードウェア資源との間に介在し、前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスとを備えた画像形成装置上で、前記アプリケーションとの整合性をチェックするバージョンチェック方法であって、前記アプリケーションのインストール時から起動後の実行時までの間に、前記アプリケーションが使用する関数単位毎のバージョン情報を取得すると共に、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得し、対応する関数のバージョンが同じか否かをチェックする関数単位バージョンチェックステップを含むことを特徴とする。

【００４４】

この請求項１６にかかる発明によれば、関数単位バージョンチェックステップによって、アプリケーションのインストール時から起動後の実行時までの間に、アプリケーションが使用する関数単位毎のバージョン情報を取得すると共に、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得して、対応する関数のバージョンが同じか否かをチェックするようにしたため、アプリケーションが使用していない関数に対応した仮想アプリケーションサービスのインターフェイスの関数が変更、追加、あるいは削除されてバージョンアップしたとしても、両者間の整合性に影響を与えないことから、全体バージョン同士を単純比較する場合と比べて、バージョン差の吸収範囲が広くなり、利用可能なアプリケーションを増やすことができる。また、アプリケーションが実際に使用する関数単位毎のバージョン同士を個別に比較するため、確実にランタイム時におけるエラー発生を防止することができる。

【００４５】

また、請求項１７にかかる発明は、請求項１６に記載のバージョンチェック方法において、前記コントロールサービスは、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うことを特徴とする。

【００４６】

この請求項１７にかかる発明によれば、コントロールサービスは、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うようにしたため、複数のアプリケーションを作成する際に、共通化できる部分をコントロールサービス（プラットホーム）として共通化することにより、それ以外の部分を作成すればよく、アプリケーションの開発が容易となり、かつ、短期間で開発することができる。

【００４７】

また、請求項１８にかかる発明は、請求項１６または１７に記載のバージョンチェック方法において、前記関数単位バージョンチェックステップによる関数単位毎のバージョンチェックを行う前に、前記各アプリケーションの全体バージョン情報を取得すると共に、前記仮想アプリケーションサービスの全体バージョン情報を取得し、全体バージョンが同じか否かをチェックする全体バージョンチェックステップをさらに含み、前記全体バージョンチェックステップによる比較結果に基づいて全体バージョンが異なっている場合のみ、前記関数単位バージョンチェックステップによる関数単位毎のバージョンチェックを行うことを特徴とする。

【００４８】

この請求項１８にかかる発明によれば、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報とを取得して、アプリケーションと仮想アプリケーションサービスの全体バージョン同士を比較する全体バージョンチェックステップをさらに含んでいて、全体バージョンが異なっている場合にのみ関数単位バージョンチェックステップによる関数単位毎のバージョンチェックを行うようにしたため、アプリケーションが使用する関数単位毎の個別のバージョンチェックを常に行う必要がなくなり、バージョンチェック処理を効率良く、かつ迅速に行うことができる。

【００４９】

また、請求項１９にかかる発明は、請求項１６～１８のいずれか一つに記載のバージョンチェック方法において、前記アプリケーションが使用する関数単位毎のバージョン情報と前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報、および、前記アプリケーションの全体バージョン情報と前記仮想アプリケーションサービスの全体バージョン情報の少なくとも一方のバージョン情報を取得するバージョン情報取得ステップをさらに含むことを特徴とする。

【００５０】

この請求項１９にかかる発明によれば、バージョン情報取得ステップによって、アプリケーションが使用する関数単位毎のバージョン情報と仮想アプリケーシ

ョンサービスのインターフェイスの関数単位毎のバージョン情報を取得したり、あるいは、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報を取得することができるため、取得したバージョン情報同士を比較するだけで、容易にバージョンチェックを行うことができる。

【００５１】

また、請求項２０にかかる発明は、請求項１９に記載のバージョンチェック方法において、前記バージョン情報取得ステップは、予め必要なバージョン情報を持たせた実行ファイルを用いて、前記アプリケーションの仮起動、あるいは、通常起動の際にプロセス間通信を行うことにより、所望のバージョン情報を取得することを特徴とする。

【００５２】

この請求項２０にかかる発明によれば、バージョン情報取得ステップは、予め必要なバージョン情報を持たせた実行ファイルを使って、アプリケーションを仮起動、あるいは、通常起動させた際のプロセス間通信により、所望のバージョン情報を取得するようにしたため、バージョン情報を容易に一元管理することができる。

【００５３】

また、請求項２１にかかる発明は、請求項１９に記載のバージョンチェック方法において、前記バージョン情報取得ステップは、前記アプリケーションの実行ファイルを生成する際に、前記アプリケーションが使用する関数単位毎のバージョン情報を記載した使用関数テーブルを生成すると共に、前記仮想アプリケーションサービスの実行ファイルを生成する際に、前記仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を記載した全関数テーブルを生成し、前記使用関数テーブルと前記全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報を取得することを特徴とする。

【００５４】

この請求項２１にかかる発明によれば、バージョン情報取得ステップは、アプリケーションの実行ファイルの生成時にアプリケーションが使用する関数毎のバ

ージョン情報が記載された使用関数テーブルを生成し、仮想アプリケーションサービスの実行ファイルの生成時に仮想アプリケーションサービスのインターフェイスの関数毎のバージョン情報が記載された全関数テーブルを生成して、使用関数テーブルと全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報が取得できるため、アプリケーションを仮起動や通常起動させることなくバージョンチェックを行うことができる。

【００５５】

また、請求項２２にかかる発明は、請求項１６～２１のいずれか一つに記載のバージョンチェック方法において、前記関数単位バージョンチェックステップは、前記アプリケーションのインストール時、前記アプリケーションの起動時、および、前記アプリケーションの起動後の実行時のいずれかの時点でバージョンチェックを行うことを特徴とする。

【００５６】

この請求項２２にかかる発明によれば、関数単位バージョンチェックステップは、アプリケーションのインストール時、アプリケーションの起動時、および、アプリケーションの起動後の実行時のいずれかの時点でバージョンチェックを行うようにしたため、アプリケーションのインストール状況等に応じて関数単位毎のバージョンチェックを適切な時期に行うことが可能となり、アプリケーションのバージョン不整合によるランタイム時のエラー発生を未然に防ぐことができる。

【００５７】

また、請求項２３にかかる発明は、請求項２２に記載のバージョンチェック方法において、前記関数単位バージョンチェックステップで前記アプリケーションのインストール時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを仮起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする。

【００５８】

この請求項２３にかかる発明によれば、関数単位バージョンチェックステップでアプリケーションのインストール時にバージョンチェックする場合、仮想アプ

リケーションサービスがアプリケーションを仮起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるため、未インストールの新規アプリケーションをインストールする際に、仮想アプリケーションサービスとの間のバージョンの整合性をチェックすることができる。このように、インストールされていないアプリケーションであっても仮起動させることにより、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信が可能となるため、各種情報を双方向でやり取りすることが可能となる。

【００５９】

また、請求項２４にかかる発明は、請求項２２に記載のバージョンチェック方法において、前記関数単位バージョンチェックステップで前記アプリケーションの起動時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを通常起動させて、当該アプリケーションの使用する関数単位毎のバージョン情報が取得できるようにしたことを特徴とする。

【００６０】

この請求項２４にかかる発明によれば、関数単位バージョンチェックステップでアプリケーションの起動時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを通常起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるため、インストール済みのアプリケーション、あるいは、未インストールのアプリケーションをとりあえずインストールしておいて、通常起動の際にバージョンチェックを受けることができる。この場合は、アプリケーションを通常起動させるため、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信を行うことが可能であり、各種情報を双方向でやり取りすることができる。

【００６１】

また、請求項２５にかかる発明は、請求項２２に記載のバージョンチェック方法において、前記関数単位バージョンチェックステップで前記アプリケーションの起動後の実行時にバージョンチェックを行う場合は、前記仮想アプリケーションサービスが前記アプリケーションを通常起動させた際に取得した、当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行され

る関数単位毎のバージョンをチェックすることを特徴とする。

【００６２】

この請求項２５にかかる発明によれば、関数単位バージョンチェックステップでアプリケーションの起動後の実行時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを通常起動させた際に取得した当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックするようにしたため、アプリケーションの起動後の実行時においてバージョンチェックを受けることができる。この場合は、既にアプリケーションを通常起動させているため、アプリケーションと仮想アプリケーションサービスとの間でのプロセス間通信が可能であり、各種情報を双方向でやり取りすることができる。

【００６３】

また、請求項２６にかかる発明は、請求項１６～２５のいずれか一つに記載のバージョンチェック方法において、前記関数単位バージョンチェックステップ、前記全体バージョンチェックステップ、および、前記バージョン情報取得ステップの少なくとも１つの処理が前記仮想アプリケーションサービスで行われることを特徴とする。

【００６４】

この請求項２６にかかる発明によれば、関数単位バージョンチェックステップ、全体バージョンチェックステップ、および、バージョン情報取得ステップの少なくとも１つの処理が仮想アプリケーションサービスで行われるため、バージョンチェック処理を主に仮想アプリケーションサービス側で行うことができる。

【００６５】

また、請求項２７にかかる発明は、請求項１６～２５のいずれか一つに記載のバージョンチェック方法において、前記関数単位バージョンチェックステップ、前記全体バージョンチェックステップ、および、前記バージョン情報取得ステップの少なくとも１つの処理が前記アプリケーションで行われることを特徴とする。

【００６６】

この請求項２７にかかる発明によれば、関数単位バージョンチェックステップ、全体バージョンチェックステップ、および、バージョン情報取得ステップの少なくとも１つの処理がアプリケーションで行われるため、バージョンチェック処理を主にアプリケーション側で行うことができる。

【００６７】

【発明の実施の形態】

以下に添付図面を参照して、この発明にかかる画像形成装置およびバージョンチェック方法の好適な実施の形態を詳細に説明する。

【００６８】

（実施の形態１）

図１は、この発明の実施の形態１である画像形成装置（以下、「複合機」という）の構成を示すブロック図である。図１に示すように、複合機１００は、白黒レーザプリンタ（B&W LP）１０１と、カラーレーザプリンタ（Color LP）１０２と、スキャナ、ファクシミリ、ハードディスク、メモリ、ネットワークインタフェースなどのハードウェアリソース１０３を有するとともに、プラットホーム１２０とアプリケーション１３０と仮想アプリケーションサービス（ＶＡＳ：Ｖｉｒｔｕａｌ　Ａｐｐｌｉｃａｔｉｏｎ　Ｓｅｒｖｉｃｅ）１４０から構成されるソフトウェア群１１０とを備えている。

【００６９】

仮想アプリケーションサービス（ＶＡＳ）１４０は、アプリケーション１３０とプラットホーム１２０の間に配置される。ＶＡＳ１４０は、アプリケーション（以下、アプリともいう）１３０の各アプリが初めて登録されるときに、同時に登録処理が行われ、アプリから見るとプラットホーム１２０のサービス層として認識され、サービス層から見るとアプリとして認識されるように登録される。このＶＡＳ１４０の基本機能としては、アプリ１３０とプラットホーム１２０との間のバージョン差を吸収して整合性を保つと共に、コントロールサービスからのメッセージを取捨選択してプラットホーム１２０を意図的に隠蔽するラッピング機能を備えている。

【００７０】

また、このＶＡＳ１４０の本発明における特徴的な機能としては、プラットホーム１２０側のバージョンアップに伴ってＶＡＳ１４０がバージョンアップされても、アプリの実行時（ランタイム時）にエラーを出さないようにするため、アプリ１３０とＶＡＳ１４０のインタフェース間のバージョンチェックを行い、動作保証の有無が事前に確認できるようにしたものである。特に、本発明では、アプリとＶＡＳの全体バージョンが違っていても、整合性のとれる限り動作保証を行うことで、ＶＡＳのサポート範囲をできるだけ広げるようにした点に特徴がある。

【００７１】

プラットホーム１２０は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一または複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ（ＳＲＭ）１２３と、汎用ＯＳ１２１とを有する。

【００７２】

コントロールサービスは、複数のサービスモジュールから形成され、ＳＣＳ（システムコントロールサービス）１２２と、ＥＣＳ（エンジンコントロールサービス）１２４と、ＭＣＳ（メモリコントロールサービス）１２５と、ＯＣＳ（オペレーションパネルコントロールサービス）１２６と、ＦＣＳ（ファックスコントロールサービス）１２７と、ＮＣＳ（ネットワークコントロールサービス）１２８とから構成される。なお、このプラットホーム１２０は、予め定義された関数により前記アプリケーション１３０からの処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）を有している。

【００７３】

汎用ＯＳ１２１は、ＵＮＩＸ（登録商標）などの汎用オペレーティングシステムであり、プラットホーム１２０並びにアプリケーション１３０の各ソフトウェアをそれぞれプロセスとして並列実行する。

【００７４】

ＳＲＭ１２３のプロセスは、ＳＣＳ１２２とともにシステムの制御およびリソースの管理を行うものである。ＳＲＭ１２３のプロセスは、スキャナ部やプリン

タ部などのエンジン、メモリ、ＨＤＤファイル、ホストＩ／Ｏ（セントロＩ／Ｆ、ネットワークＩ／Ｆ、ＩＥＥＥ１３９４　Ｉ／Ｆ、ＲＳ２３２Ｃ　Ｉ／Ｆなど）のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行制御する。

【００７５】

具体的には、このＳＲＭ１２３は、要求されたハードウェア資源の利用が可能であるか（他の要求により利用されていないかどうか）を判断し、利用可能であれば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、ＳＲＭ１２３は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

【００７６】

ＳＣＳ１２２のプロセスは、アプリ管理、操作部制御、システム画面表示、ＬＥＤ表示、リソース管理、割り込みアプリ制御などを行う。

【００７７】

ＥＣＳ１２４のプロセスは、白黒レーザプリンタ（B&W LP）１０１、カラーレーザプリンタ（Color LP）１０２、スキャナ、ファクシミリなどからなるハードウェアリソース１０３のエンジンの制御を行う。

【００７８】

ＭＣＳ１２５のプロセスは、画像メモリの取得および解放、ハードディスク装置（ＨＤＤ）の利用、画像データの圧縮および伸張などを行う。

【００７９】

ＦＣＳ１２７のプロセスは、システムコントローラの各アプリ層からＰＳＴＮ／ＩＳＤＮ網を利用したファクシミリ送受信、ＢＫＭ（バックアップＳＲＡＭ）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのＡＰＩを提供する。

【００８０】

ＮＣＳ１２８のプロセスは、ネットワークＩ／Ｏを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネット

ワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデータをネットワーク側に送信する際の仲介を行う。具体的には、ftpd、httpd、lpd、snmpd、telnetd、smtpdなどのサーバデーモンや、同プロトコルのクライアント機能などを有している。

【００８１】

ＯＣＳ１２６のプロセスは、オペレータ（ユーザ）と本体制御間の情報伝達手段となるオペレーションパネル（操作パネル）の制御を行う。ＯＣＳ１２６は、オペレーションパネルからキー押下をキーイベントとして取得し、取得したキーに対応したキーイベント関数をＳＣＳ１２２に送信するＯＣＳプロセスの部分と、アプリケーション１３０またはコントロールサービスからの要求によりオペレーションパネルに各種画面を描画出力する描画関数やその他オペレーションパネルに対する制御を行う関数などが予め登録されたＯＣＳライブラリの部分とから構成される。このＯＣＳライブラリは、アプリケーション１３０およびコントロールサービスの各モジュールにリンクされて実装されている。なお、ＯＣＳ１２６のすべてをプロセスとして動作させるように構成しても良く、あるいはＯＣＳ１２６のすべてをＯＣＳライブラリとして構成しても良い。

【００８２】

アプリケーション１３０は、ページ記述言語（ＰＤＬ）、ＰＣＬおよびポストスクリプト（ＰＳ）を有するプリンタ用のアプリケーションであるプリンタアプリ１１１と、コピー用アプリケーションであるコピーアプリ１１２と、ファクシミリ用アプリケーションであるファックスアプリ１１３と、スキャナ用アプリケーションであるスキャナアプリ１１４と、ネットワークファイル用アプリケーションであるネットファイルアプリ１１５と、工程検査用アプリケーションである工程検査アプリ１１６とを有している。これらの各アプリは、その起動時にＶＡＳ１４０に対して自プロセスのプロセスＩＤとともにアプリ登録要求メッセージを送信し、アプリ登録要求メッセージを受信したＶＡＳ１４０によって、起動したアプリに対する登録が行われるようになっている。

【００８３】

アプリケーション１３０の各プロセス、コントロールサービスの各プロセスは

、関数呼び出しとその戻り値送信およびメッセージの送受信によってプロセス間通信を行いながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを実現している。

【００８４】

このように、実施の形態１にかかる複合機１００には、複数のアプリケーション１３０および複数のコントロールサービスが存在し、いずれもプロセスとして動作している。そして、これらの各プロセス内部には、一または複数のスレッドが生成されて、スレッド単位の並列実行が行われる。そして、コントロールサービスがアプリケーション１３０に対し共通サービスを提供しており、このため、これらの多数のプロセスが並列動作、およびスレッドの並列動作を行って互いにプロセス間通信を行って協調動作をしながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを提供するようになっている。また、複合機１００には、サードベンダなどの第三者がコントロールサービス層の上のアプリケーション層に新規アプリ１１７、１１８を開発して搭載することが可能となっている。図１では、この新規アプリ１１７、１１８を搭載した例を示している。

【００８５】

なお、実施の形態１にかかる複合機１００では、複数のアプリケーション１３０のプロセスと複数のコントロールサービスのプロセスとが動作しているが、アプリケーション１３０とコントロールサービスのプロセスをそれぞれ単一の構成とすることも可能である。また、各アプリケーション１３０は、アプリケーションごとに追加または削除することができる。

【００８６】

図２は、実施の形態１にかかる複合機１００のＶＡＳ１４０の構成と、ＶＡＳ１４０と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図２では、アプリケーション１３０の例として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７、１１８を示しているが、他のアプリでも同様の構成である。

【００８７】

仮想アプリケーションサービス（ＶＡＳ）１４０のプロセスには、ディスパッチャ１４５と、制御スレッド１４４と、バージョン情報取得スレッド１４３と、全体バージョンチェックスレッド１４２と、関数単位バージョンチェックスレッド１４１とが動作している。

【００８８】

ディスパッチャ１４５は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて制御スレッド１４４、バージョン情報取得スレッド１４３、全体バージョンチェックスレッド１４２、関数単位バージョンチェックスレッド１４１に処理要求を行うものである。実施の形態１の複合機１００では、ディスパッチャ１４５は、ＶＡＳ１４０によって各アプリを仮起動させたり、通常起動したときにアプリとの間で行われるプロセス間通信を、制御スレッド１４４を介して各スレッド１４１～１４３間でやり取りすることができる。

【００８９】

制御スレッド１４４は、ディスパッチャ１４４を介して送られてくる各アプリが使用する関数単位毎のバージョン情報や各アプリの全体バージョン情報などをバージョン情報取得スレッド１４３、全体バージョンチェックスレッド１４２、関数単位バージョンチェックスレッド１４１にそれぞれ渡したり、各スレッド１４１～１４３間における処理順序を制御したり、各スレッド１４１～１４３からの処理要求をディスパッチャ１４５に伝えたりする。

【００９０】

また、関数単位バージョンチェック手段としての関数単位バージョンチェックスレッド１４１は、アプリが使用する関数単位毎のバージョン情報と、ＶＡＳ１４０のインタフェースの全関数単位毎のバージョン情報とを取得して、対応する関数同士のバージョンを比較して、同じか否かをチェックするものである。

【００９１】

図３は、実施の形態１におけるアプリとＶＡＳのインタフェースとの間のバージョンチェック状態を説明する図である。図３に示すように、アプリとＶＡＳをメイクする場合は、それぞれのソースプログラムをコンパイルしてオブジェクト

プログラムとする。そして、両者が共通に使用するインタフェースの関数をインクルードファイル（図中のａｐｉｖｅｒｓｉｏｎ．ｈ３０１）とし、オブジェクトプログラムの先頭部分にインクルードして、そのオブジェクトプログラムをリンクさせることにより、実行形式のプログラムを作成することができる。このため、アプリとＶＡＳのヘッダに組み込まれているインタフェースの関数（ここでは、ＡＰＩ）の全体バージョンは、当初（１．００）と同じである。

【００９２】

しかし、本発明では、バージョンチェックを全体バージョン同士を単純比較するのではなく、関数単位（ＡＰＩ単位）のバージョンを比較し、かつ、アプリが使用している関数についてのみ比較することにより、全体バージョンが違っても動作保証ができる場合がある点に着目している（バージョンの変更や追加がアプリの使用していないＡＰＩに関するものの場合）。すなわち、これを図３で見ると、アプリ側から取得するバージョン情報を当該アプリのみが使用する関数単位毎のバージョン情報（図中の使用ＡＰＩバージョン３０２）としたので、ＶＡＳの全関数単位毎のバージョン情報（図中の全ＡＰＩバージョン３０３）がバージョンアップして、全体バージョンも（１．００）→（１．０１）となり（図中の全ＡＰＩバージョン３０４参照）、関数単位のＡＰＩ　Ｎｏ．＝２５１のバージョンが（１０１）→（１０２）に変更されても、アプリが使用していない関数のバージョンであるため、整合性には影響はなく、動作保証を得ることができる。このため、全体バージョン同士を単純比較して、バージョン差がある場合は全て動作保証が得られないとする従来のバージョンチェックと比べると、サポート範囲を広げることが可能となり、ＶＡＳの再吸収範囲を実質的に拡張することができる。

【００９３】

また、図２に示す全体バージョンチェック手段としての全体バージョンチェックスレッド１４２は、各アプリ毎の全体バージョン情報と、ＶＡＳ１４０のインタフェースの全体バージョン情報とを取得して、全体バージョン同士が同じか否かをチェックするものである。この手段は従来と同じであるが、本発明の特徴である関数単位毎のバージョンチェックの前段処理として、まず全体バージョン同

士を比較して、全体バージョン同士が同じであれば整合性が有りと言えるので、関数単位毎のバージョンチェックを行う必要がなくなり、処理を簡略化することができる。そして、上記した関数単位毎のバージョンチェックは、全体バージョンが違っている場合にのみ行うようにする。

【００９４】

さらに、図２に示すバージョン情報取得手段としてのバージョン情報取得スレッド１４３は、上記した関数単位バージョンチェックスレッド１４１で必要なアプリが使用する関数単位毎のバージョン情報とＶＡＳのインターフェイスの関数単位毎のバージョン情報を取得したり、上記した全体バージョンチェックスレッド１４２で必要なアプリの全体バージョン情報とＶＡＳのインタフェースの全体バージョン情報を取得するものである。このバージョン情報取得スレッド１４３には、種々の形態が考えられ、新規アプリのインストール時にバージョンチェックしようとすると、まだアプリが起動されていないためプロセス間通信などを使って情報を取得することができない。このため、ＶＡＳ１４０が対象となるアプリを仮起動させて、プロセス間通信によりアプリから必要な情報を取得するものである。ここでは、少なくとも当該アプリが使用する関数単位毎のバージョン情報を取得したが、これ以外にもプロダクトＩＤ（ベンダー、アプリ、バージョンから一義的に決定される）、ベンダー名、アプリケーション名、全体バージョン、リソース情報等を併せて取得することも可能である。また、対象となるアプリが格納されるインストール対象エリアとしては、ＨＤＤ、ＩＣカード、各種メモリカード、あるいは、ネットワークに接続されたパーソナルコンピュータ(ＰＣ）などがある。

【００９５】

図４は、アプリケーションを仮起動処理するか通常起動処理するかを引数によって指定するメイン関数記述例の図である。上記したように、ＶＡＳ１４０がアプリを起動する際に、通常起動と仮起動とを容易に使い分けることができる。例えば、図４に示すように、引数ｉ（ｉオプション）を使って指定すると、仮起動がなされて、アプリ情報通知がなされる。また、引数ｉ（ｉオプション）を用いない場合は、通常起動を行って、アプリ本来の動作を行うことができる。勿論、

後述するアプリ起動時、あるいは、アプリ起動後の実行時にバージョンチェックする場合は、この通常起動処理が用いられる。

【００９６】

また、バージョン情報取得スレッド１４３は、上記以外にソースプログラムのアプリとＶＡＳをコンパイル／リンクして実行ファイルを作成する際に、少なくとも関数単位バージョンチェックスレッド１４１が必要とするアプリの使用する関数単位毎のバージョン情報をテーブル化した使用関数テーブル２１２をＶＡＳ１４０へ通知したり、ＶＡＳ１４０のインタフェースの全関数のバージョン情報をテーブル化した全関数テーブル２１１をＲＡＭ２１０などのメモリ上に作成し、必要に応じて利用できるようにすることが考えられる。

【００９７】

次に、動作について説明する。図５は、アプリインストール時における仮起動のシーケンスを説明するフローチャートであり、図６は、図５の中のバージョンチェック処理のサブルーチンを示すフローチャートである。

【００９８】

まず、図５に示すように、ユーザがオペレーションパネル上のキーやボタンを使ってアプリのインストール処理を選択すると（ステップＳ５０１）、ＳＣＳ（システムコントロールサービス）１２２からＶＡＳ１４０にアプリインストール処理の開始を要求し（ステップＳ５０２）、ＶＡＳ１４０がインストール対象エリア内のアプリを仮起動させる（ステップＳ５０３）。この仮起動は、インストールするアプリ内の各種情報をプロセス間通信を使ってＶＡＳ１４０に通知させ、本発明のバージョンチェック処理を行うものである（ステップＳ５０４）。

【００９９】

ステップＳ５０４におけるバージョンチェック処理のサブルーチンは、図６に示すように、アプリから通知された情報にアプリの全体バージョン情報が含まれているか否かを判断し（ステップＳ６０１）、全体バージョン情報が含まれている場合は、ＶＡＳ１４０のインタフェースの全体バージョンと比較し、同じであれば（ステップＳ６０２）、バージョンの不整合はなく、ＶＡＳ１４０のサポート範囲内として動作保証する（ステップＳ６０３）。

【０１００】

また、図６のステップＳ６０１およびステップＳ６０２おいて、それぞれＮＯが選択された場合は、ステップＳ６０４に移行し、アプリが使用する関数に対応したＶＡＳ１４０のインタフェースの関数毎のバージョンが全て同じか否かが判断され、同じであれば上記ステップＳ６０３にてＶＡＳ１４０のサポート範囲内として動作保証する（ステップＳ６０３）。しかしステップＳ６０４でバージョンの異なる関数があった場合は、ＶＡＳ１４０のサポート範囲外となるため、動作保証できない（ステップＳ６０５）。

【０１０１】

このように、図５のステップＳ５０４のサブルーチンでバージョンチェック処理を行った後、他にインストール処理を行うアプリが無くなるまで上記ステップＳ５０３に戻り、上記バージョンチェック処理が繰り返される（ステップＳ５０５）。

【０１０２】

インストール処理するアプリが無くなった場合は、ステップＳ５０６でインストール時に取得したアプリの情報に基づいてインストール画面をオペレーションパネル上などに生成し、各種情報を表示する（ステップＳ５０６）。例えば、インストール処理された複数のアプリのバージョンチェック結果に基づいて動作保証の有無を表示すれば、ユーザは、正式なインストール要求すべきアプリをインストール画面から容易に選択することができ（ステップＳ５０７）、選択されたアプリのみを最終的にインストール処理する（ステップ５０８）。

【０１０３】

上記したように、実施の形態１では、アプリのインストール時に本発明のバージョンチェック処理を行う場合は、ＶＡＳによってアプリを仮起動させ、プロセス間通信を利用してバージョン情報を含むアプリに関する情報を取得して、バージョンチェック行うことができるので、動作保証されたアプリのみを選択的にインストール処理することができる。

【０１０４】

また、上記以外にもアプリの通常起動時を利用して本発明のバージョンチェッ

ク処理を行うことができる。アプリを仮起動させるか通常起動させるかは、図４のメイン関数記述例に示したように、引数ｉ（ｉオプション）を用いるか否かで決まり、通常起動時にも勿論プロセス間通信を使ってアプリのバージョン情報などが取得できるため、バージョンチェック処理を行うことができる。このアプリの起動時におけるバージョンチェックは、既にインストールされているアプリを初めて起動する場合などに有効である。

【０１０５】

さらに、実施の形態１では、上記以外に、アプリ起動後の実行時にも本発明のバージョンチェック処理を行うことができる。動作保証のないアプリを実行するとランタイム時にエラーが発生する可能性があるため、遅くとも実行時までには本発明のバージョンチェックを行う必要がある。アプリ起動後の実行時には、プロセス間通信が使えるため、アプリの通常起動時におけるバージョンチェックと同じように行っても良いが、プロセス間通信は時間がかかるため、スムーズな実行処理が行えない。

【０１０６】

そこで、実施の形態１では、以下の図７および図８のフローチャートを使った別の動作例について説明する。図７は、アプリとＶＡＳをメイクする際にバージョンチェックに必要な関数テーブルを生成する手順を説明するフローチャートであり、図８は、図７で生成した関数テーブルを使ってアプリの実行時にバージョンチェックを行う手順を説明するフローチャートである。

【０１０７】

まず、図７のステップＳ７０１では、アプリとＶＡＳを新たに作成して画像形成装置に搭載することを要する。そして、作成するアプリとＶＡＳが共通に使用するＡＰＩをインクルードファイルとして独立させておく（ステップＳ７０２）。図３に示したａｐｉｖｅｒｓｉｏｎ．ｈ３０１がこれに相当する。

【０１０８】

そして、アプリとＶＡＳのそれぞれのソースプログラムを別々にコンパイルしてオブジェクトファイルとし（ステップＳ７０３）、そのオブジェクトファイルの先頭部分に上記したインクルードファイルをインクルードする（ステップＳ７

０４）。さらに、インクルードファイルが組み込まれたアプリとＶＡＳのオブジェクトファイルとをリンクすることにより、実行ファイルが作成される（ステップＳ７０５）。

【０１０９】

その後、アプリ側では、自分の使用する関数毎のバージョンが記載された使用関数テーブル２１２を生成し、アプリ内部に格納しておく（ステップＳ７０６）。また、ＶＡＳ側では、そのインタフェースの全関数毎のバージョンが記載された全関数テーブル２１１を生成して、ＶＡＳ内部あるいはＲＡＭ２１０などのメモリに格納しておく（ステップＳ７０７）。

【０１１０】

これに続く図８のフローチャートでは、図７のステップＳ７０６，Ｓ７０７で生成された使用関数テーブル２１２と全関数テーブル２１１とを使って、アプリの実行時にバージョンチェックを行っている。まず、図８では、アプリとＶＡＳのメイク後にいずれかのバージョンの更新があったか否かが問われ（ステップＳ８０１）、更新があった場合はテーブルを新たに生成し直すことにより（ステップＳ８０２）、常に最新のバージョン情報が反映された関数テーブルを維時することができる。

【０１１１】

そして、ＶＡＳ上でアプリを実行する場合は（ステップＳ８０３）、アプリの使用関数テーブル２１２の内容をＶＡＳへ通知する（ステップＳ８０４）。この通知は、情報ファイルの通知としてアプリからＶＡＳへ一方的に送られるもので、プロセス間通信と比べると格段に早く、アプリの実行時に与える影響を小さくすることができる。

【０１１２】

ＶＡＳ１４０は、アプリから通知される使用関数テーブル２１２のバージョン情報に基づいて、対応する関数単位毎のバージョンを比較し、全てのバージョンが同じか否かを判断する（ステップＳ８０５）。関数単位毎のバージョンが全て同じであれば、バージョンの不整合はなく、ＶＡＳ１４０のサポート範囲内であるとして動作保証する（ステップＳ８０６）。また、ステップＳ８０５でバージ

ョンの異なる関数があった場合は、ＶＡＳ１４０のサポート範囲外となるため、動作保証されない（ステップＳ８０７）。ここでは、アプリの実行時にバージョンチェックを行っているため、図８のフローチャートでは詳細に説明していないが、ステップＳ８０６で動作保証有りと判断された場合は、アプリの実行がそのまま継続して行われる。また、ステップＳ８０７で動作保証無しと判断された場合は、アプリの実行処理を中止し、オペレーションパネル上にエラー表示などを行って、ユーザにその旨を通知するなどの対処法が考えられる。

【０１１３】

このように、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッドにより、アプリが使用する関数単位毎のバージョン情報とＶＡＳのインターフェースの関数単位毎のバージョン情報を取得し、対応する関数同士のバージョンが同じか否かをチェックするので、アプリが使用していないＶＡＳのインタフェースの関数が変更、追加、あるいは削除されて、全体のバージョンが変わっても、両者間の整合性に影響を与えないため、全体バージョン同士を単純比較する場合と比べると、ＶＡＳのバージョン差の吸収範囲が広くなって、利用可能なアプリを増やすことができる。また、アプリのインストール時から起動後の実行時までの間にバージョンチェックを行うため、ランタイム時におけるエラー発生を確実に防止することができる。

【０１１４】

また、実施の形態１にかかる複合機では、アプリとＶＡＳの全体バージョン同士を比較する全体バージョンチェックスレッドをさらに備えているため、全体バージョンが異なっている場合のみ関数単位バージョンチェックスレッドにより関数単位毎のバージョンチェックをすればよいので、常に関数単位毎のバージョンチェックを行う必要がなく、バージョンチェック処理を効率良く、かつ迅速に行うことができる。

【０１１５】

また、実施の形態１にかかる複合機では、バージョン情報取得スレッドにより、アプリが使用する関数単位毎のバージョン情報とＶＡＳのインターフェイスの関数単位毎のバージョン情報を取得したり、アプリの全体バージョン情報とＶＡ

Ｓの全体バージョン情報を取得することが可能なため、取得したバージョン情報同士を比較するだけで、容易にバージョンチェックを行うことができる。

【０１１６】

また、実施の形態１にかかる複合機では、バージョン情報取得スレッドにより、予め必要なバージョン情報を持たせた実行ファイルを使い、アプリの仮起動や通常起動の際のプロセス間通信により、所望のバージョン情報を取得することができるので、バージョン情報を容易に一元管理することができる。

【０１１７】

また、実施の形態１にかかる複合機では、バージョン情報取得スレッドは、アプリの実行ファイルの生成時にアプリが使用する関数毎のバージョン情報が記載された使用関数テーブルを生成し、ＶＡＳの実行ファイルの生成時にＶＡＳのインターフェイスの関数毎のバージョン情報が記載された全関数テーブルを生成して、使用関数テーブルと全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報を取得することができるため、アプリケーションを仮起動や通常起動せずにバージョンチェックすることができる。

【０１１８】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッドは、アプリのインストール時、アプリの起動時、アプリの起動後の実行時のいずれかの時点でバージョンチェックを行うようにしたため、アプリのインストール状況等に応じて関数単位毎のバージョンチェックを適切な時期に行うことが可能となり、アプリのバージョン不整合によるランタイム時のエラーの発生を未然に防ぐことができる。

【０１１９】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッドがアプリのインストール時にバージョンチェックする場合、ＶＡＳがアプリを仮起動させて、アプリが使用する関数単位毎のバージョン情報を得ることができるので、新規アプリケーションをインストールする際にＶＡＳとの間のバージョンの整合性をチェックすることができる。

【０１２０】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッドがアプリの起動時にバージョンチェックする場合、ＶＡＳがアプリを通常起動して、そのアプリが使用する関数単位毎のバージョン情報が得られるので、アプリとＶＡＳとの間のバージョンの整合性をチェックすることができる。

【０１２１】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッドがアプリの起動後の実行時にバージョンチェックする場合、ＶＡＳがアプリを通常起動した際に取得した当該アプリが使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックするため、アプリの起動後の実行時にバージョンチェックを行うことができる。

【０１２２】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッド、全体バージョンチェックスレッド、および、バージョン情報取得スレッドの少なくとも１つがＶＡＳに含まれるようにしたため、バージョンチェック処理を主にＶＡＳ側で行うことができる。

【０１２３】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッド、全体バージョンチェックスレッド、および、バージョン情報取得スレッドの少なくとも１つがＶＡＳのプロセス内部でスレッドとして生成されるため、複数のアプリを並列に起動したり、バージョン情報取得処理と関数単位バージョンチェック処理と全体バージョンチェック処理との並列処理を円滑に行うことができる。

【０１２４】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッド、全体バージョンチェックスレッド、および、バージョン情報取得スレッドの少なくとも１つがアプリケーションに含まれるようにしたため、バージョンチェック処理を主にアプリ側で行うことができる。

【０１２５】

また、実施の形態１にかかる複合機では、関数単位バージョンチェックスレッド、全体バージョンチェックスレッド、および、バージョン情報取得スレッドの少なくとも１つがアプリケーションのプロセス内部でスレッドとして生成されるので、これらの並列処理を円滑に行うことができる。

【０１２６】

（実施の形態２）

実施の形態１にかかる複合機１００は、ＶＡＳ１４０が全アプリケーションに対して１つのみ存在するものであった、この実施の形態２にかかる複合機では、各アプリごとに一つのＶＡＳが起動し、各ＶＡＳは対応するアプリとの間でバージョンチェック処理などが行われる。

【０１２７】

図９は、実施の形態２にかかる複合機９００の構成を示すブロック図である。図９に示すように、複合機９００では、複数の仮想アプリケーションサービス（ＶＡＳ）９４１～９４８がアプリケーション１３０の各アプリごとに動作している点が、実施の形態１にかかる複合機１００と異なっている。

【０１２８】

ＶＡＳ９４１～９４８は、プリンタアプリ１１１、コピーアプリ１１２、ファックスアプリ１１３、スキャナアプリ１１４、ネットファイルアプリ１１５、工程検査アプリ１１６、新規アプリ１１７および１１８に対応して、バージョンチェック処理を行うようになっている。

【０１２９】

図１０は、実施の形態２にかかる複合機９００のＶＡＳ９４１～９４８の構成と、ＶＡＳ９４１～９４８と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図１０では、アプリケーション１３０として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７、１１８の例を示し、さらにこれら各アプリに対応したＶＡＳ９４１、９４２、９４７および９４８を例として示しているが、他のアプリの場合も同様の構成である。

【０１３０】

また、実施の形態２にかかる複合機９００では、実施の形態１の複合機１００と異なり、図１０に示すように、各ＶＡＳ９４１～９４８と各アプリとの間にはＶＡＳ制御プロセス（デーモン）９０１が動作している。

【０１３１】

このＶＡＳ制御プロセス（デーモン）９０１は、各アプリとＶＡＳが共通に使用するＡＰＩをインクルードファイルとして持っていて、アプリとＶＡＳのソースプログラムを別々にコンパイルしてオブジェクトファイルとし、その先頭部分にインクルードファイルをインクルードする。そして、このアプリとＶＡＳのオブジェクトファイルをリンクさせて対応した実行ファイルを作成する。

【０１３２】

仮想アプリケーションサービス（ＶＡＳ）９４１～９４８のプロセスには、ディスパッチャ１４５と、関数単位バージョンチェックスレッド１４１と、全体バージョンチェックスレッド１４２と、バージョン情報取得スレッド１４３とが動作している。

【０１３３】

ディスパッチャ１４５は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて関数単位バージョンチェックスレッド１４１、全体バージョンチェックスレッド１４２、バージョン情報取得スレッド１４３に処理要求を行うものである。実施の形態２の複合機９００では、ディスパッチャ１４５は、各アプリを仮起動したり通常起動したときに、アプリとの間で行われるプロセス間通信を、ＶＡＳ制御プロセス９０１を介して各ＶＡＳ９４１～９４８のスレッド１４１～１４３との間でやり取りすることができる。

【０１３４】

関数単位バージョンチェックスレッド１４１は、アプリが使用する関数単位毎のバージョン情報と、ＶＡＳ１４０のインタフェースの全関数単位毎のバージョン情報の対応する関数同士のバージョンを比較して、同じか否かをチェックするものである。

【０１３５】

全体バージョンチェックスレッド１４２は、各アプリ毎の全体バージョン情報と、ＶＡＳ１４０のインタフェースの全体バージョン情報とを取得して、全体バージョン同士が同じか否かをチェックするものである。

【０１３６】

バージョン情報取得スレッド１４３は、上記した関数単位バージョンチェックスレッド１４１で必要なアプリが使用する関数単位毎のバージョン情報とＶＡＳのインターフェイスの関数単位毎のバージョン情報を取得したり、上記した全体バージョンチェックスレッド１４２で必要なアプリの全体バージョン情報とＶＡＳのインタフェースの全体バージョン情報を取得するものである。

【０１３７】

このように、実施の形態２にかかる複合機９００によれば、実施の形態１にかかる複合機１００と同様に、アプリが使用していない関数に対応したＶＡＳのインターフェイスの関数が変更、追加、あるいは削除されてバージョンアップしたとしても、両者間の整合性に影響を与えないことから、全体バージョン同士を単純比較する場合と比べて、バージョン差の吸収範囲が広くなり、利用可能なアプリケーションを増やすことができる。その上、アプリケーションが実際に使用する関数単位毎のバージョン同士を個別に比較するため、ランタイム時におけるエラー発生を確実に防止することができる。

【０１３８】

また、実施の形態２にかかる複合機９００では、ＶＡＳ９４１～９４８がアプリ毎に設けられ、各アプリを個別に仮起動または通常起動させることができるため、各アプリが使用する関数単位毎のバージョン情報を取得すると共に、対応するＶＡＳのインターフェイスの関数単位毎のバージョン情報を取得し、対応した関数同士のバージョンチェックを容易に行うことができる。

【０１３９】

【発明の効果】

以上説明したように、請求項１にかかる発明によれば、関数単位バージョンチェック手段によって、アプリケーションのインストール時から起動後の実行時までの間に、アプリケーションが使用する関数単位毎のバージョン情報を取得する

と共に、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得して、対応する関数のバージョンが同じか否かをチェックするようにしたので、アプリケーションが使用していない関数に対応した仮想アプリケーションサービスのインターフェイスの関数が変更、追加、あるいは削除されてバージョンアップしたとしても、両者間の整合性に影響を与えないことから、全体バージョン同士を単純比較する場合と比べて、バージョン差の吸収範囲が広くなり、利用可能なアプリケーションを増やすことができる。また、アプリケーションが実際に使用する関数単位毎のバージョン同士を個別に比較するので、確実にランタイム時におけるエラー発生を防止することができる。

【０１４０】

また、請求項２にかかかる発明によれば、仮想アプリケーションサービスは、アプリケーション毎に設けられ、各アプリケーションを個別に仮起動または通常起動させるようにしたので、各アプリケーションが使用する関数単位毎のバージョン情報を取得し、これに対応した仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得して、対応する関数同士のバージョンチェックを容易に行うことができる。

【０１４１】

また、請求項３にかかる発明によれば、コントロールサービスは、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うようにしたので、複数のアプリケーションを作成する際に、共通化できる部分をコントロールサービス（プラットホーム）として共通化することにより、それ以外の部分を作成すればよく、アプリケーションの開発が容易となり、かつ、短期間で開発することができる。

【０１４２】

また、請求項４にかかる発明によれば、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報とを取得して、アプリ

ケーションと仮想アプリケーションサービスの全体バージョン同士を比較する全体バージョンチェック手段をさらに備えていて、全体バージョンが異なっている場合にのみ関数単位バージョンチェック手段による関数単位毎のバージョンチェックを行うようにしたので、アプリケーションが使用する関数単位毎の個別のバージョンチェックを常に行う必要がなくなり、バージョンチェック処理を効率良く、かつ迅速に行うことができる。

【０１４３】

また、請求項５にかかる発明によれば、バージョン情報取得手段により、アプリケーションが使用する関数単位毎のバージョン情報と仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得したり、あるいは、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報を取得することができるので、取得したバージョン情報同士を比較するだけで、容易にバージョンチェックを行うことができる。

【０１４４】

また、請求項６にかかる発明によれば、バージョン情報取得手段は、予め必要なバージョン情報を持たせた実行ファイルを使って、アプリケーションを仮起動、あるいは、通常起動させた際のプロセス間通信により、所望のバージョン情報を取得するようにしたので、バージョン情報を容易に一元管理することができる。

【０１４５】

また、請求項７にかかる発明によれば、バージョン情報取得手段は、アプリケーションの実行ファイルの生成時にアプリケーションが使用する関数毎のバージョン情報が記載された使用関数テーブルを生成し、仮想アプリケーションサービスの実行ファイルの生成時に仮想アプリケーションサービスのインターフェイスの関数毎のバージョン情報が記載された全関数テーブルを生成して、使用関数テーブルと全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報が取得できるので、アプリケーションを仮起動や通常起動させることなくバージョンチェックを行うことができる。

【０１４６】

また、請求項８にかかる発明によれば、関数単位バージョンチェック手段は、アプリケーションのインストール時、アプリケーションの起動時、および、アプリケーションの起動後の実行時のいずれかの時点でバージョンチェックを行うようにしたので、アプリケーションのインストール状況等に応じて関数単位毎のバージョンチェックを適切な時期に行うことが可能となり、アプリケーションのバージョン不整合によるランタイム時のエラー発生を未然に防ぐことができる。

【０１４７】

また、請求項９にかかる発明によれば、関数単位バージョンチェック手段がアプリケーションのインストール時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを仮起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるので、未インストールの新規アプリケーションをインストールする際に、仮想アプリケーションサービスとの間のバージョンの整合性をチェックすることができる。このように、インストールされていないアプリケーションであっても仮起動させることにより、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信が可能となるので、各種情報を双方向でやり取りすることが可能となる。

【０１４８】

また、請求項１０にかかる発明によれば、関数単位バージョンチェック手段がアプリケーションの起動時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを通常起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるので、インストール済みのアプリケーション、あるいは、未インストールのアプリケーションをとりあえずインストールしておいて、通常起動の際にバージョンチェックを受けることができる。この場合は、アプリケーションを通常起動させるので、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信を行うことが可能であり、各種情報を双方向でやり取りすることができる。

【０１４９】

また、請求項１１にかかる発明によれば、関数単位バージョンチェック手段がアプリケーションの起動後の実行時にバージョンチェックする場合、仮想アプリ

ケーションサービスがアプリケーションを通常起動させた際に取得した当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックするようにしたので、アプリケーションの起動後の実行時においてバージョンチェックを受けることができる。この場合は、既にアプリケーションを通常起動させているので、アプリケーションと仮想アプリケーションサービスとの間でのプロセス間通信が可能であり、各種情報を双方向でやり取りすることができる。

【０１５０】

また、請求項１２にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つが仮想アプリケーションサービスに含まれているので、バージョンチェック処理を主に仮想アプリケーションサービス側で行うことができる。

【０１５１】

また、請求項１３にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つが仮想アプリケーションサービスのプロセス内部でスレッドとして生成されるので、複数のアプリケーションが並列に起動されたり、バージョン情報取得処理と関数単位バージョンチェック処理と全体バージョンチェック処理とを、コンテクスト切り替えなしにＣＰＵ占有切り替えによる並列実行を行うことが可能となり、これらの並列処理を円滑に行うことができる。

【０１５２】

また、請求項１４にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つがアプリケーションに含まれているので、バージョンチェック処理を主にアプリケーション側で行うことができる。

【０１５３】

また、請求項１５にかかる発明によれば、関数単位バージョンチェック手段、全体バージョンチェック手段、および、バージョン情報取得手段の少なくとも１つがアプリケーションのプロセス内部でスレッドとして生成されるので、バージ

ョン情報取得処理と関数単位バージョンチェック処理と全体バージョンチェック処理とを、コンテクスト切り替えなしにＣＰＵ占有切り替えによる並列実行を行うことが可能となり、これらの並列処理を円滑に行うことができる。

【０１５４】

また、請求項１６にかかる発明によれば、関数単位バージョンチェックステップによって、アプリケーションのインストール時から起動後の実行時までの間に、アプリケーションが使用する関数単位毎のバージョン情報を取得すると共に、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得して、対応する関数のバージョンが同じか否かをチェックするようにしたので、アプリケーションが使用していない関数に対応した仮想アプリケーションサービスのインターフェイスの関数が変更、追加、あるいは削除されてバージョンアップしたとしても、両者間の整合性に影響を与えないことから、全体バージョン同士を単純比較する場合と比べて、バージョン差の吸収範囲が広くなり、利用可能なアプリケーションを増やすことができる。また、アプリケーションが実際に使用する関数単位毎のバージョン同士を個別に比較するので、確実にランタイム時におけるエラー発生を防止することができる。

【０１５５】

また、請求項１７にかかる発明によれば、コントロールサービスは、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うようにしたので、複数のアプリケーションを作成する際に、共通化できる部分をコントロールサービス（プラットホーム）として共通化することにより、それ以外の部分を作成すればよく、アプリケーションの開発が容易となり、かつ、短期間で開発することができる。

【０１５６】

また、請求項１８にかかる発明によれば、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報とを取得して、アプ

リケーションと仮想アプリケーションサービスの全体バージョン同士を比較する全体バージョンチェックステップをさらに含んでいて、全体バージョンが異なっている場合にのみ関数単位バージョンチェックステップによる関数単位毎のバージョンチェックを行うようにしたので、アプリケーションが使用する関数単位毎の個別のバージョンチェックを常に行う必要がなくなり、バージョンチェック処理を効率良く、かつ迅速に行うことができる。

【０１５７】

また、請求項１９にかかる発明によれば、バージョン情報取得ステップによって、アプリケーションが使用する関数単位毎のバージョン情報と仮想アプリケーションサービスのインターフェイスの関数単位毎のバージョン情報を取得したり、あるいは、アプリケーションの全体バージョン情報と仮想アプリケーションサービスの全体バージョン情報を取得することができるので、取得したバージョン情報同士を比較するだけで、容易にバージョンチェックを行うことができる。

【０１５８】

また、請求項２０にかかる発明によれば、バージョン情報取得ステップは、予め必要なバージョン情報を持たせた実行ファイルを使って、アプリケーションを仮起動、あるいは、通常起動させた際のプロセス間通信により、所望のバージョン情報を取得するようにしたので、バージョン情報を容易に一元管理することができる。

【０１５９】

また、請求項２１にかかる発明によれば、バージョン情報取得ステップは、アプリケーションの実行ファイルの生成時にアプリケーションが使用する関数毎のバージョン情報が記載された使用関数テーブルを生成し、仮想アプリケーションサービスの実行ファイルの生成時に仮想アプリケーションサービスのインターフェイスの関数毎のバージョン情報が記載された全関数テーブルを生成して、使用関数テーブルと全関数テーブルのいずれか一方のバージョン情報を他方へ通知することにより、比較可能なバージョン情報が取得できるので、アプリケーションを仮起動や通常起動させることなくバージョンチェックを行うことができる。

【０１６０】

また、請求項２２にかかる発明によれば、関数単位バージョンチェックステップは、アプリケーションのインストール時、アプリケーションの起動時、および、アプリケーションの起動後の実行時のいずれかの時点でバージョンチェックを行うようにしたので、アプリケーションのインストール状況等に応じて関数単位毎のバージョンチェックを適切な時期に行うことが可能となり、アプリケーションのバージョン不整合によるランタイム時のエラー発生を未然に防ぐことができる。

【０１６１】

また、請求項２３にかかる発明によれば、関数単位バージョンチェックステップでアプリケーションのインストール時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを仮起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるので、未インストールの新規アプリケーションをインストールする際に、仮想アプリケーションサービスとの間のバージョンの整合性をチェックすることができる。このように、インストールされていないアプリケーションであっても仮起動させることにより、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信が可能となるので、各種情報を双方向でやり取りすることが可能となる。

【０１６２】

また、請求項２４にかかる発明によれば、関数単位バージョンチェックステップでアプリケーションの起動時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを通常起動させて、そのアプリケーションが使用する関数単位毎のバージョン情報を得ることができるので、インストール済みのアプリケーション、あるいは、未インストールのアプリケーションをとりあえずインストールしておいて、通常起動の際にバージョンチェックを受けることができる。この場合は、アプリケーションを通常起動させるので、アプリケーションと仮想アプリケーションサービスとの間でプロセス間通信を行うことが可能であり、各種情報を双方向でやり取りすることができる。

【０１６３】

また、請求項２５にかかる発明によれば、関数単位バージョンチェックステッ

プでアプリケーションの起動後の実行時にバージョンチェックする場合、仮想アプリケーションサービスがアプリケーションを通常起動させた際に取得した当該アプリケーションの使用する関数単位毎のバージョン情報に基づいて、その後実行される関数単位毎のバージョンをチェックするようにしたので、アプリケーションの起動後の実行時においてバージョンチェックを受けることができる。この場合は、既にアプリケーションを通常起動させているので、アプリケーションと仮想アプリケーションサービスとの間でのプロセス間通信が可能であり、各種情報を双方向でやり取りすることができる。

【０１６４】

また、請求項２６にかかる発明によれば、関数単位バージョンチェックステップ、全体バージョンチェックステップ、および、バージョン情報取得ステップの少なくとも１つの処理が仮想アプリケーションサービスで行われるので、バージョンチェック処理を主に仮想アプリケーションサービス側で行うことができる。

【０１６５】

また、請求項２７にかかる発明によれば、関数単位バージョンチェックステップ、全体バージョンチェックステップ、および、バージョン情報取得ステップの少なくとも１つの処理がアプリケーションで行われるので、バージョンチェック処理を主にアプリケーション側で行うことができる。

【図面の簡単な説明】

【図１】

この発明の実施の形態１である画像形成装置の構成を示すブロック図である。

【図２】

実施の形態１にかかる複合機のＶＡＳの構成と各アプリとコントロールサービス層と汎用ＯＳとの関係を示すブロック図である。

【図３】

実施の形態１におけるアプリとＶＡＳのインタフェースとの間のバージョンチェック状態を説明する図である。

【図４】

アプリケーションを仮起動処理するか通常起動処理するかを引数によって指定

するメイン関数記述例の図である。

【図５】

アプリインストール時における仮起動のシーケンスを説明するフローチャートである。

【図６】

図５の中のバージョンチェック処理のサブルーチンを示すフローチャートである。

【図７】

アプリとＶＡＳをメイクする際にバージョンチェックに必要な関数テーブルを生成する手順を説明するフローチャートである。

【図８】

図７で生成した関数テーブルを使ってアプリの実行時にバージョンチェックを行う手順を説明するフローチャートである。

【図９】

実施の形態２にかかる複合機の構成を示すブロック図である。

【図１０】

実施の形態２にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。

【符号の説明】

１００　複合機

１０１　白黒レーザプリンタ

１０２　カラーレーザプリンタ

１０３　ハードウェアリソース

１１０　ソフトウェア群

１１１　プリンタアプリ

１１２　コピーアプリ

１１３　ファックスアプリ

１１４　スキャナアプリ

１１５　ネットファイルアプリ

１１６　工程検査アプリ

１１７、１１８　新規アプリ

１２０　プラットホーム

１２１　汎用ＯＳ

１２２　ＳＣＳ

１２３　ＳＲＭ

１２４　ＥＣＳ

１２５　ＭＣＳ

１２６　ＯＣＳ

１２７　ＦＣＳ

１２８　ＮＣＳ

１３０　アプリケーション

１４０、９４１～９４８　仮想アプリケーションサービス（ＶＡＳ）

１４１　関数単位バージョンチェックスレッド

１４２　全体バージョンチェックスレッド

１４３　バージョン情報取得スレッド

１４４　制御スレッド

１４５　ディスパッチャ

１５０　コントロールサービス層

２０１　ラッピング処理情報ファイル

２１０　ＲＡＭ

２１１　全関数テーブル

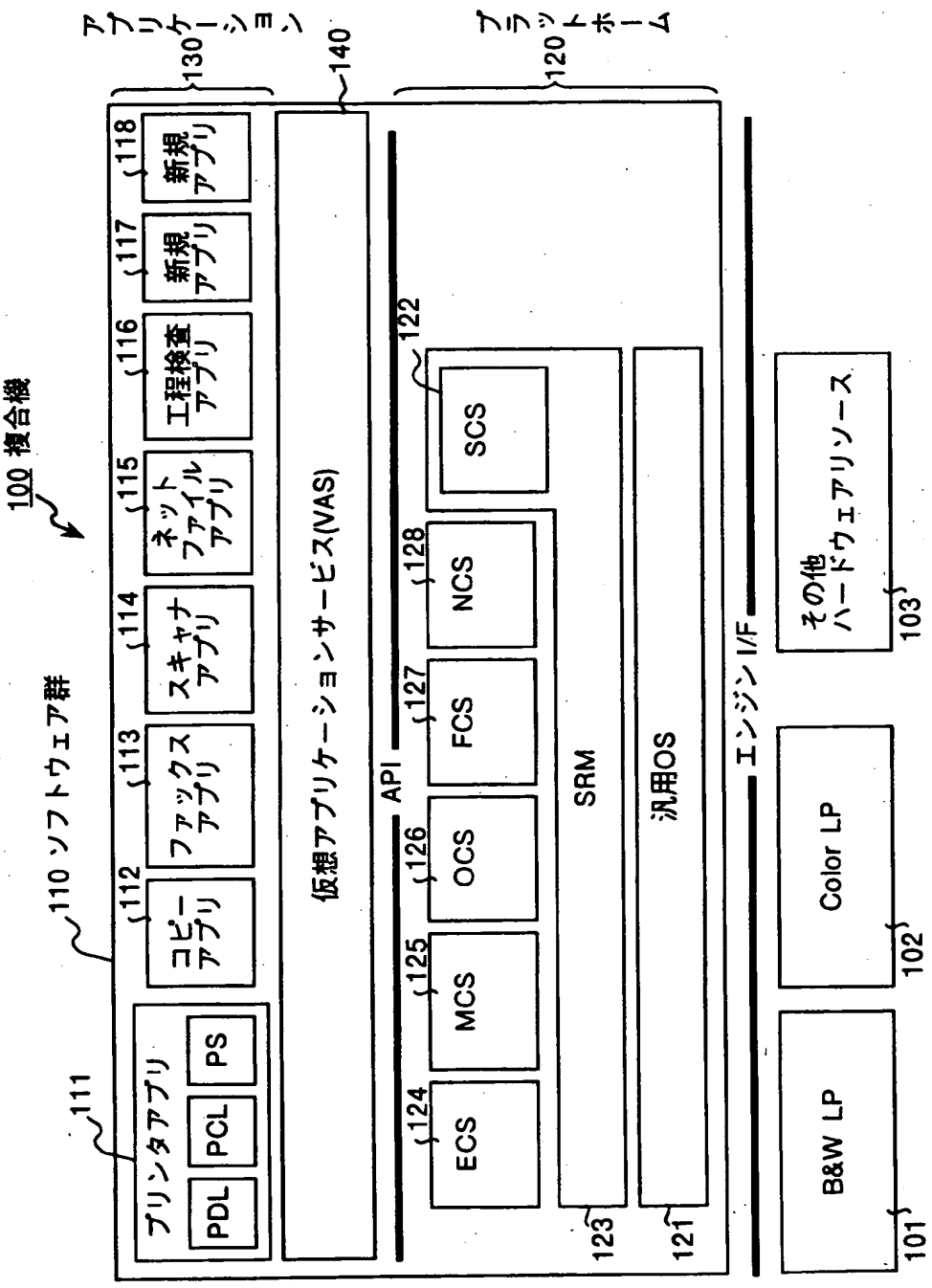
２１２　使用関数テーブル

９００　複合機

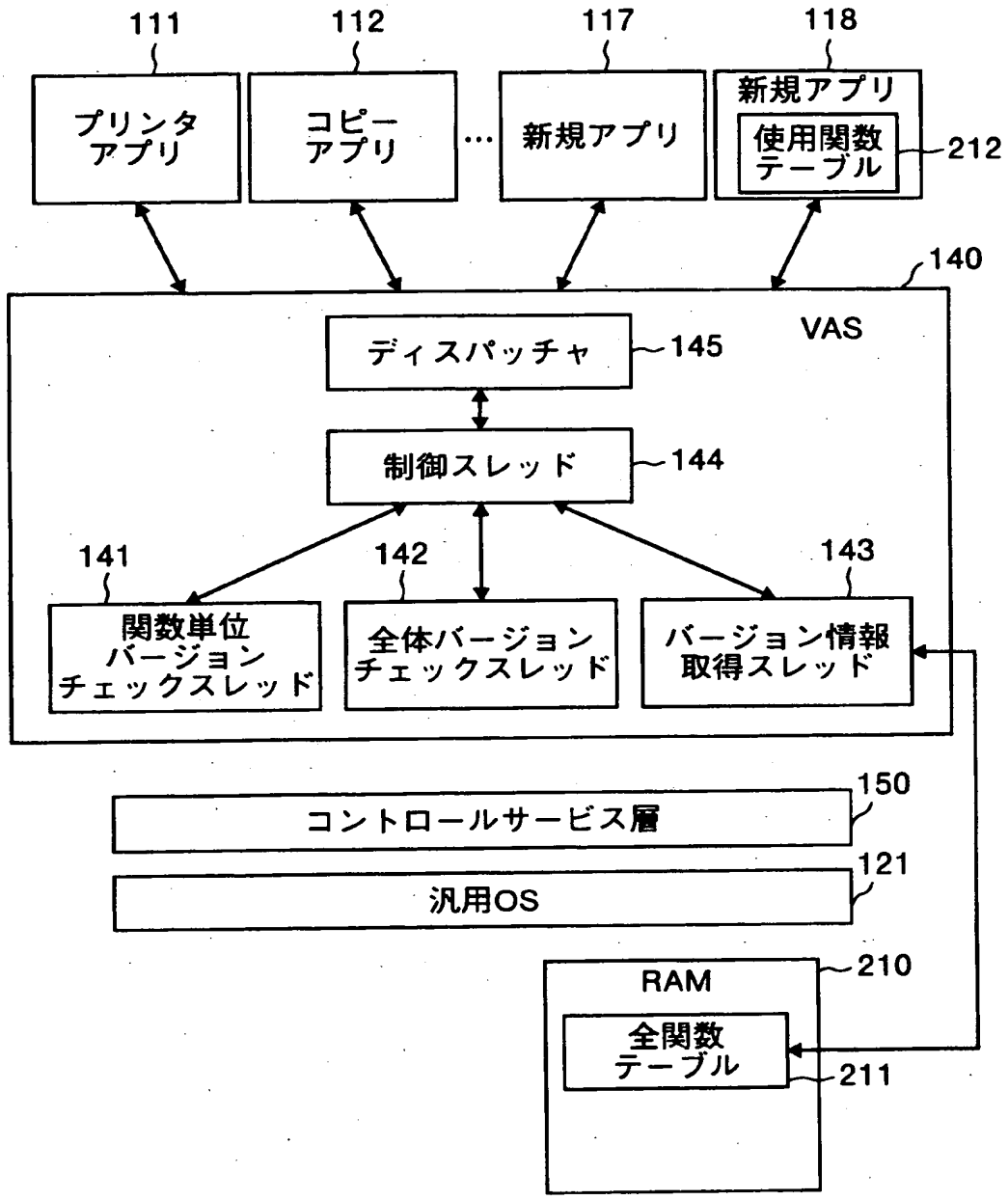
９０１　ＶＡＳ制御プロセス

【書類名】　　図面

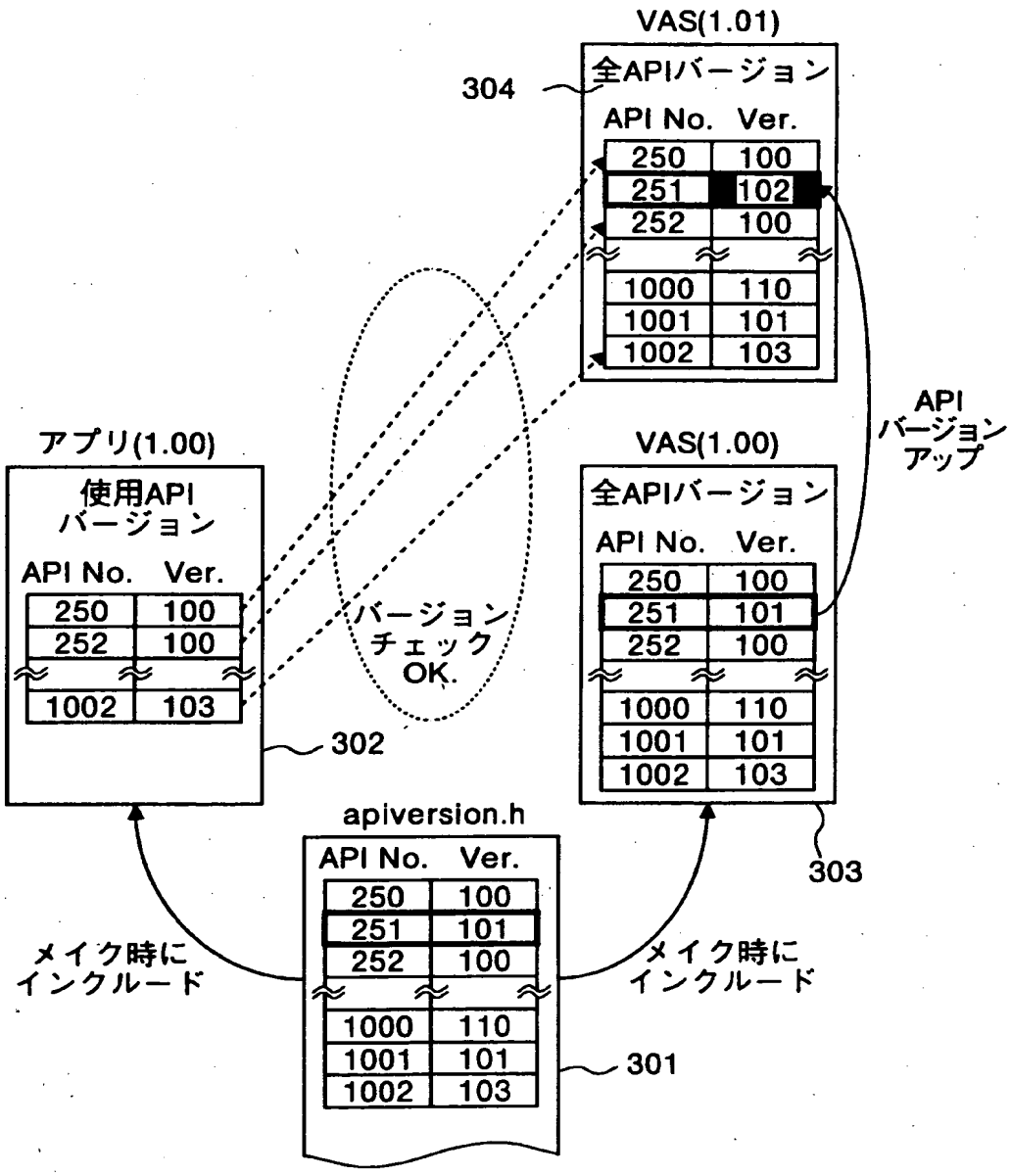
【図１】


アプリケーション 130
プラットホーム 120
140
100 複合機
110 ソフトウェア群
111
プリンタアプリ
PDL
PCL
PS
112
コピーアプリ
113
ファックスアプリ
114
スキャナアプリ
115
ネットファイルアプリ
116
工程検査アプリ
117
新規アプリ
118
新規アプリ
仮想アプリケーションサービス(VAS)
API
122
SCS
128
NCS
127
FCS
126
OCS
125
MCS
124
ECS
SRM
123
汎用OS
121
エンジン I/F
その他ハードウェアリソース
103
Color LP
102
B&W LP
101

【図２】


111
112
117
118
プリンタ
アプリ
コピー
アプリ
新規アプリ
新規アプリ
使用関数
テーブル
212
140
VAS
ディスパッチャ
145
制御スレッド
144
141
142
143
関数単位
バージョン
チェックスレッド
全体バージョン
チェックスレッド
バージョン情報
取得スレッド
150
コントロールサービス層
121
汎用OS
RAM
210
全関数
テーブル
211

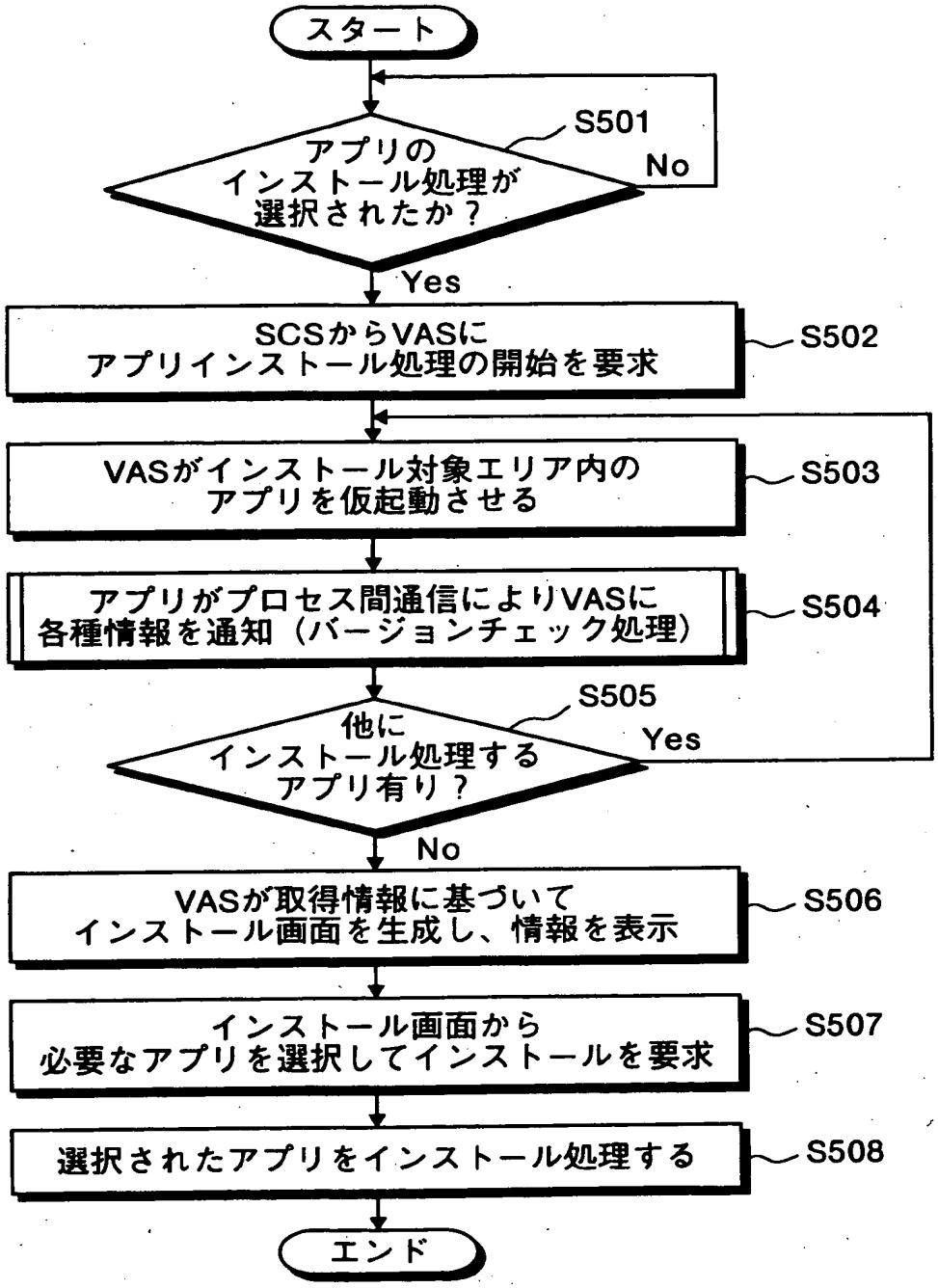
【図３】


VAS(1.01)
全APIバージョン
304
API No. Ver.
250 100
251 102
252 100
1000 110
1001 101
1002 103
APIバージョンアップ
アプリ(1.00)
使用API バージョン
API No. Ver.
250 100
252 100
1002 103
302
バージョンチェックOK.
VAS(1.00)
全APIバージョン
API No. Ver.
250 100
251 101
252 100
1000 110
1001 101
1002 103
303
apiversion.h
API No. Ver.
250 100
251 101
252 100
1000 110
1001 101
1002 103
301
メイク時にインクルード
メイク時にインクルード

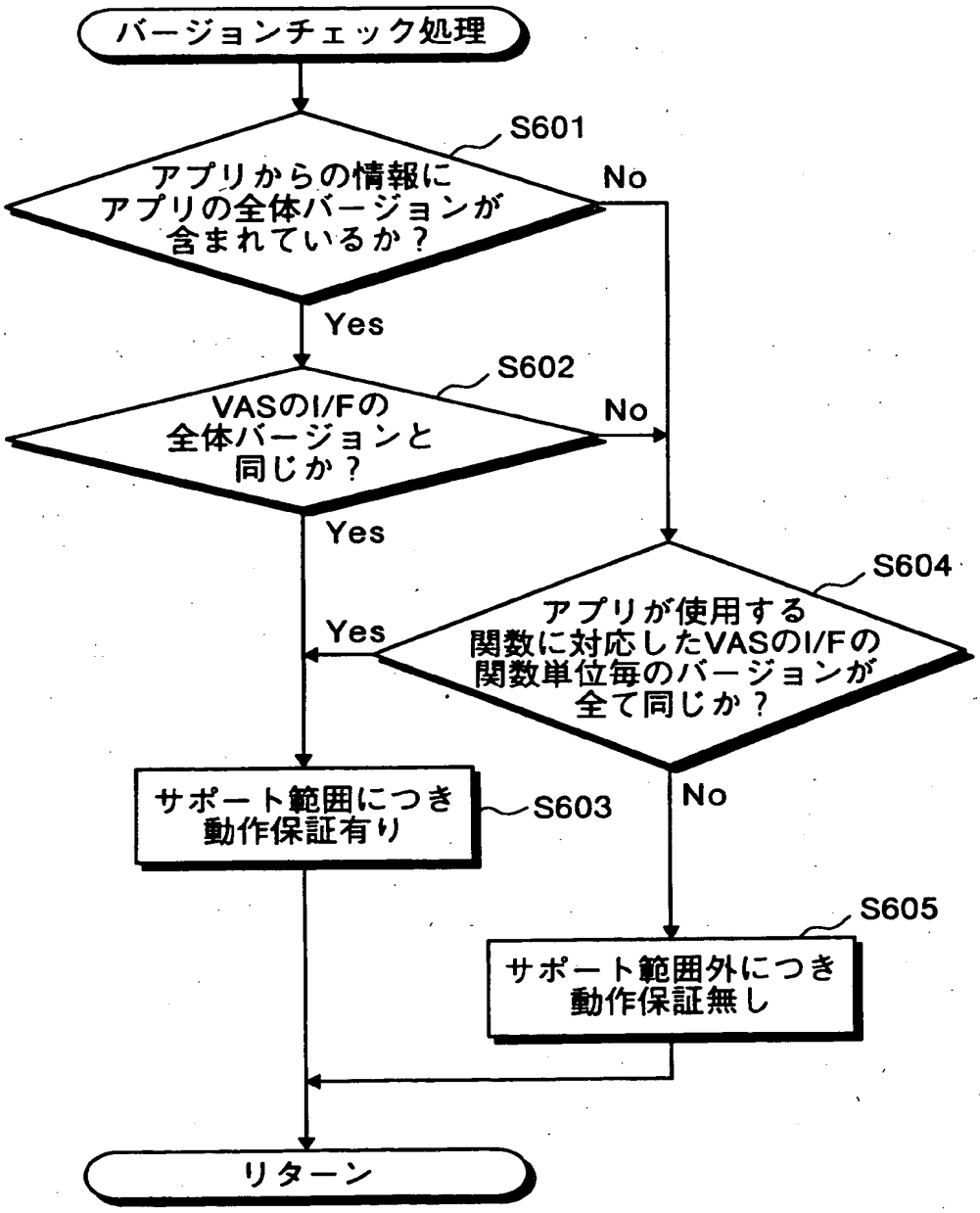
【図４】

```
void main(int argc,char **argv)
{
        if( (argc>1)&&f(argv[1]=='i') ) {
                仮起動処理 (アプリ情報通知)
                exit(1);
        } else {
                通常起動処理 (アプリ本来の動作)
        }
}
```

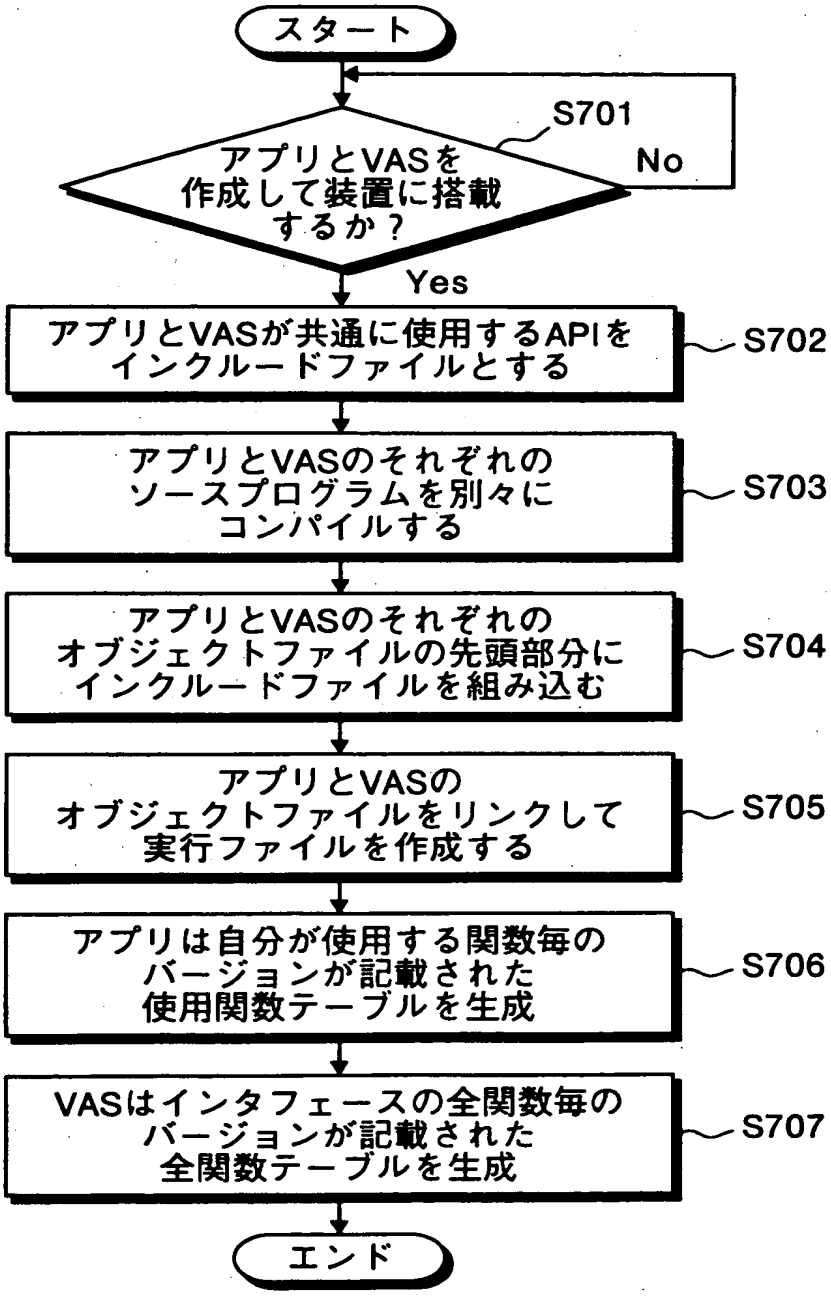
【図５】


スタート
S501
アプリの
インストール処理が
選択されたか？
No
Yes
SCSからVASに
アプリインストール処理の開始を要求
S502
VASがインストール対象エリア内の
アプリを仮起動させる
S503
アプリがプロセス間通信によりVASに
各種情報を通知（バージョンチェック処理）
S504
S505
他に
インストール処理する
アプリ有り？
Yes
No
VASが取得情報に基づいて
インストール画面を生成し、情報を表示
S506
インストール画面から
必要なアプリを選択してインストールを要求
S507
選択されたアプリをインストール処理する
S508
エンド

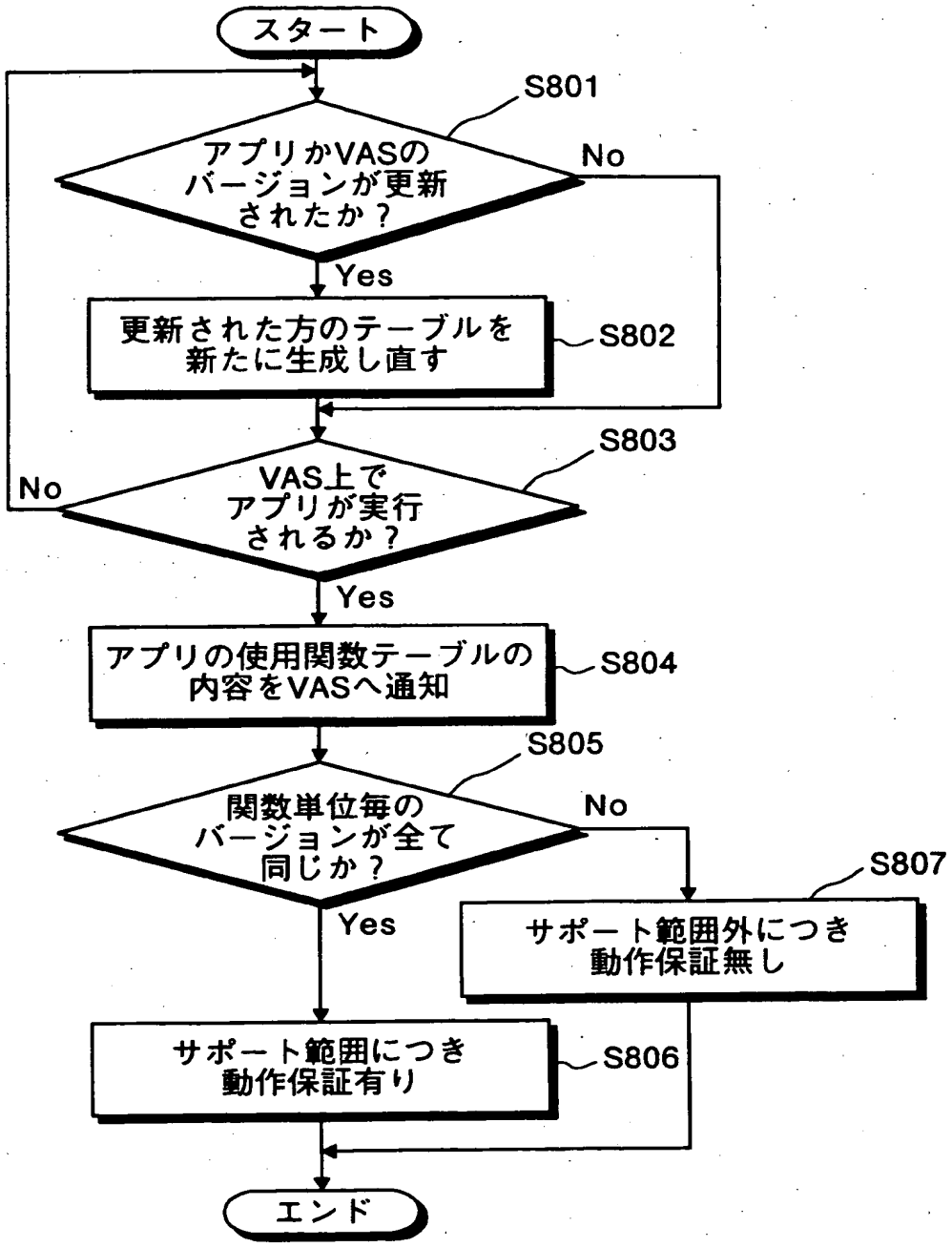
【図６】


バージョンチェック処理
S601
アプリからの情報に
アプリの全体バージョンが
含まれているか？
No
Yes
S602
VASのI/Fの
全体バージョンと
同じか？
No
Yes
S604
アプリが使用する
関数に対応したVASのI/Fの
関数単位毎のバージョンが
全て同じか？
Yes
No
サポート範囲につき
動作保証有り
S603
S605
サポート範囲外につき
動作保証無し
リターン

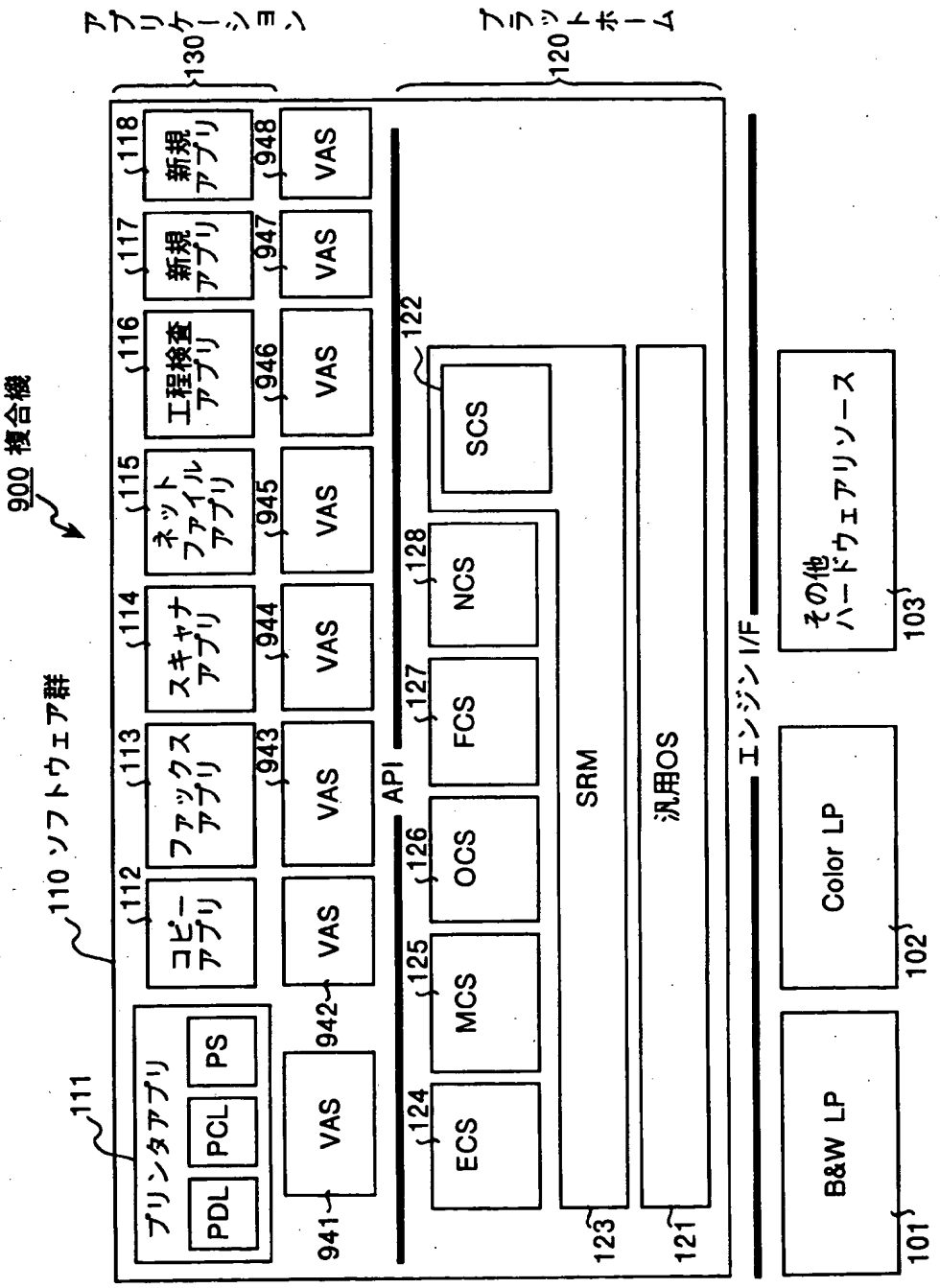
【図７】


スタート
S701
アプリとVASを
作成して装置に搭載
するか？
No
Yes
アプリとVASが共通に使用するAPIを
インクルードファイルとする
S702
アプリとVASのそれぞれの
ソースプログラムを別々に
コンパイルする
S703
アプリとVASのそれぞれの
オブジェクトファイルの先頭部分に
インクルードファイルを組み込む
S704
アプリとVASの
オブジェクトファイルをリンクして
実行ファイルを作成する
S705
アプリは自分が使用する関数毎の
バージョンが記載された
使用関数テーブルを生成
S706
VASはインタフェースの全関数毎の
バージョンが記載された
全関数テーブルを生成
S707
エンド

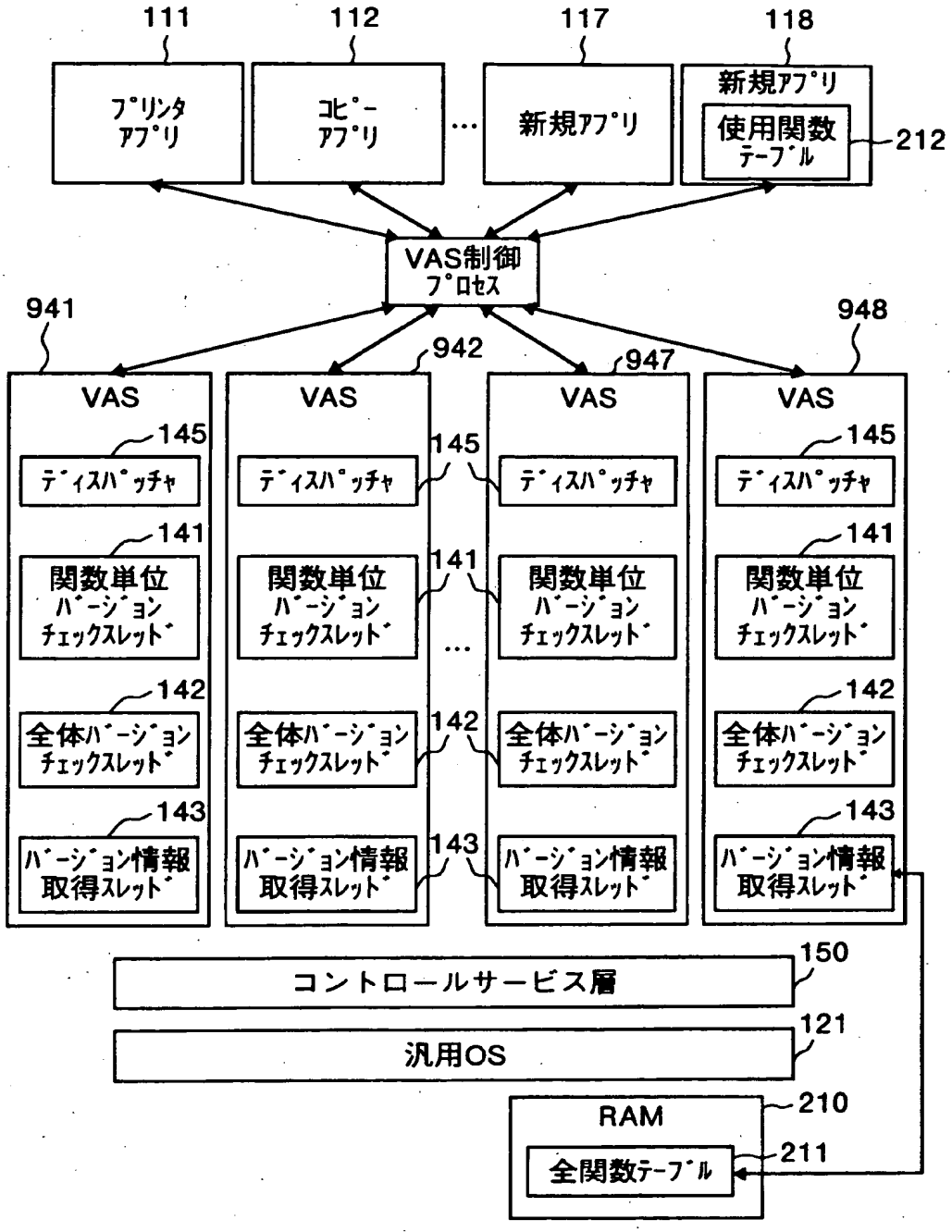
【図８】


スタート
S801
アプリかVASの
バージョンが更新
されたか？
No
Yes
更新された方のテーブルを
新たに生成し直す
S802
S803
No
VAS上で
アプリが実行
されるか？
Yes
アプリの使用関数テーブルの
内容をVASへ通知
S804
S805
関数単位毎の
バージョンが全て
同じか？
No
S807
Yes
サポート範囲外につき
動作保証無し
サポート範囲につき
動作保証有り
S806
エンド

【図９】


アプリケーション
130
プラットホーム
120
900 複合機
110 ソフトウェア群
111
プリンタアプリ
PDL
PCL
PS
112
コピーアプリ
113
ファックスアプリ
114
スキャナアプリ
115
ネットファイルアプリ
116
工程検査アプリ
117
新規アプリ
118
新規アプリ
941
942
943
944
945
946
947
948
VAS
API
122
SCS
128
NCS
127
FCS
126
OCS
125
MCS
124
ECS
123
SRM
121
汎用OS
エンジンI/F
B&W LP
101
Color LP
102
その他ハードウェアリソース
103

【図１０】


111
112
117
118
ﾌﾟﾘﾝﾀ ｱﾌﾟﾘ
ｺﾋﾟｰ ｱﾌﾟﾘ
新規ｱﾌﾟﾘ
新規ｱﾌﾟﾘ
使用関数 ﾃｰﾌﾞﾙ
212
VAS制御 ﾌﾟﾛｾｽ
941
942
947
948
VAS
145
ﾃﾞｨｽﾊﾟｯﾁｬ
141
関数単位 ﾊﾞｰｼﾞｮﾝ ﾁｪｯｸｽﾚｯﾄﾞ
142
全体ﾊﾞｰｼﾞｮﾝ ﾁｪｯｸｽﾚｯﾄﾞ
143
ﾊﾞｰｼﾞｮﾝ情報 取得ｽﾚｯﾄﾞ
150
コントロールサービス層
121
汎用OS
RAM
210
211
全関数ﾃｰﾌﾞﾙ

【書類名】　　要約書

【要約】

【課題】　アプリと仮想アプリケーションサービスとの間のバージョン差をできるだけ吸収して利用可能なアプリを増やし、ランタイム時のエラー発生を確実に防止するようにする。

【解決手段】　新規アプリ１１８のインストール時に、ＶＡＳ１４０が新規アプリ１１８を仮起動させて、プロセス間通信を使うことにより、必要な情報を取得することができる。ここでは、新規アプリ１１８が使用する関数単位毎のバージョン情報をバージョン情報取得スレッド１４３で取得し、関数単位バージョンチェックスレッド１４１によって、ＶＡＳ１４０のインタフェースの関数単位毎のバージョン情報と比較して、アプリとＶＡＳとの間のバージョン差の吸収範囲を広げることができる。

【選択図】　　図２

出　願　人　履　歴　情　報

識別番号　　〔０００００６７４７〕

１．変更年月日　　２００２年　５月１７日
［変更理由］　住所変更
住　所　　東京都大田区中馬込１丁目３番６号
氏　名　　株式会社リコー